Bibliography

Cloninger C R, Sigvardsson S, Bohman M 1996 Type I and type II alcoholism: An update. *Alcohol Health and Research World* **20**: 18–23

Crabbe J C, Wahlsten D, Dudek B C 1999 Genetics of mouse behavior: Interactions with laboratory environment. *Science* **284**: 1670–1

Feldman R S, Meyer J S, Quenzer L F 1997 *Principles of Neuropsychopharmacology*. Sinauer Associates

Ferguson R A, Goldberg D M 1997 Genetic markers of alcohol abuse. *Clinica Chimica Acta* **257**: 199–250

Frankel W N, Schork N J 1996 Who's afraid of epistasis? *Nature Genetics* **14**: 371–3

Goate A M, Edenberg J H 1998 The genetics of alcoholism. *Current Opinion in Genetics and Development* B: 282–6

Long J C, Knowler W C, Hanson R L, Robin R W, Urbanek M, Moore E, Bennett P H, Goldman D 1998 Evidence for genetic linkage to alcohol dependence on chromosomes 4 and 11 from an autosome-wide scan in an American Indian population. *American Journal of Medical Genetics* **81**: 216–21

Morey L C 1996 Patient placement criteria: Linking typologies to managed care. *Alcohol Health and Research World* **20**: 36–44

Vaillant G E 1983 *The Natural History of Alcoholism*. Harvard University Press, Cambridge, MA

K. E. Browman and J. C. Crabbe

Algorithmic Complexity

1. Introduction

In the mid 1960s, in the early stage of computer science but with the general theory of Turing machines (Turing 1936) well understood, scientists needed to measure computation and information quantitatively. Kolmogorov complexity was invented by R. J. Solomonoff (1964), A. N. Kolmogorov (1965), and G. J. Chaitin (1969), independently and in this chronological order. This theory is now widely accepted as the standard approach that settled a half-century debate about the notion of randomness of an individual object—as opposed to the better understood notion of a random variable with intuitively both 'random' and 'nonrandom' individual outcomes. Kolmogorov complexity has a plethora of applications in many areas including computer science, mathematics, physics, biology, and social sciences (Li and Vitányi 1993). This article only describes some basic ideas and some appropriate sample applications.

Intuitively, the amount of information in a finite string is the size (number of bits) of the smallest program that, started with a blank memory, computes the string and then terminates. A similar definition can be given for infinite strings, but in this case the program produces element after element forever. Thus, 1^n (a string of n ones) contains little information because a program of size about $\log n$ outputs it. Likewise, the transcendental number $\pi = 3.1415\ldots$, an infinite sequence of seemingly 'random' decimal digits, contains a constant amount ($O(1)$) of information. (There is a short program that produces the consecutive digits of π forever.) Such a definition would appear to make the amount of information in an object depend on the particular programming language used. This is the case. Fortunately it can be shown that all choices of universal programming languages (such as PASCAL, C^{++}, Java, or LISP in which we can in principle program every task that can intuitively be programmed at all) lead to quantification of the amount of information that is invariant up to an additive constant. Formally, it is best formulated in terms of 'universal Turing machines,' the celebrated rigorous formulation of 'computability' by A. M. Turing (1936) that started both the theory and practice of computation.

This theory is different from Shannon information theory that deals with the expected information in a message from a probabilistic ensemble of possible messages. Kolmogorov complexity, on the other hand, measures the information in an individual string or message. The *randomness deficiency* of a binary string n bits long is the number of bits by which the complexity falls short of n—the maximum complexity—and a string is the more random the closer the complexity is to its length.

2. Theory

The Kolmogorov complexity $C(x)$ of a string x is the length of the shortest binary program (for a fixed reference universal programming language) that prints x as its only output and then halts. A string x is incompressible if $C(x)$ is at least the length $|x|$ (number of bits) of x: the shortest way to describe x is to give it literally. Similarly, a string x is 'nearly' incompressible if $C(x)$ is 'almost as large as' $|x|$.

The appropriate standard for 'almost as large' above can depend on the context, a typical choice being $C(x) \geqslant |x| - O(|\log x|)$. Similarly, the *conditional* Kolmogorov complexity of x with respect to y, denoted by $C(x|y)$, is the length of the shortest binary program that, with extra information y, prints x. And a string x is incompressible relative to y if $C(x|y)$ is large in the appropriate sense.

Intuitively, we think of such patternless sequences as being random, and we use the term 'random sequence' synonymously with 'incompressible sequence.' This is not just a matter of naming but on the contrary embodies the resolution of the fundamental question about the existence and characterization of random individual objects (strings). Following a half-century of unsuccessful approaches and acrimonious scientific debates, in 1965 the Swedish mathematician Per Martin-Löf resolved the matter and gave a rigorous formalization of the intuitive notion of a random sequence as a sequence that passes all effective tests for randomness. He gave a similar formulation for infinite random sequences. The set of infinite random sequences has measure 1 in the set of all sequences. Martin-Löf's formulation uses constructive measure theory and has equivalent formulations in terms of being incompressible. Every Martin-Löf random sequence is *universally* random in the sense that it individually possesses all effectively testable randomness properties. (One can compare this with the notion of intuitive computability that is precisely captured by the notion of 'computable by Turing machines,' and every Turing machine computation can be performed by a universal Turing machine.) Many applications depend on the following easy facts.

Lemma 1 *Let c be a positive integer. For every fixed y, every finite set A contains at least* $(1-2^{-c})|A|+1$ *elements x with* $C(x|A, y) \geqslant [\log|A|]-c$. (*Choosing A to be the set of all strings of length n we have* $C(x/n,y) \geqslant n-c$)

Lemma 2 *Let A be a finite set. For every y, every element* $x \in A$ *has complexity* $C(x|A, y) \leqslant \log|A|+c$. (*Choosing A to be the set of all strings of length n we have* $C(x/n,y) \leqslant n+c$)

The first lemma is proved by simple counting. The second lemma holds since a fixed program that enumerates the given finite set computes x from its index in the enumeration order—and this index has $\log|A|$ bits for a set A of cardinality $|A|$.

We can now compare Kolmogorov complexity with Shannon's statistical notion of entropy—the minimal *expected* code word length of messages from random source using the most parsimonious code possible. Surprisingly, many laws that hold for Shannon entropy (that is, on average) still hold for the Kolmogorov complexity of individual strings albeit only within a logarithmic additive term. Denote by $C(x|y)$ the information in x given y (the length of the shortest program that computes x from y), and denote by $C(x, y)$ the length of the shortest program that computes the pair x, y. Here is the (deep and powerful) Kolmogorov complexity version of the classical 'symmetry of information' law. Up to an additive logarithmic term,

$$C(x, y) = C(x)+C(y|x) = C(y)+C(x|y) \qquad (1)$$

We can interpret $C(x)-C(x|y)$ as the information y has about x. It follows from the above that the amount of information y has about x is almost the same as the amount of information x has about y: information is symmetric. This is called mutual information.

Kolmogorov complexity is a wonderful measure of randomness. However, it is not computable, which obviously impedes some forms of practical use. Nevertheless, noncomputability is not really an obstacle for the wide range of applications of Kolmogorov complexity, just like noncomputability of almost all real numbers does not impede their practical ubiquitous use.

3. *Applications*

For numerous applications in computer science, combinatorics, mathematics, learning theory, philosophy, biology, and physics, see Li and Vitányi (1993). For illustrative applications in cognitive psychology see Chater (1996) (related, more informal strains of thought are the Structural Information Theory started in Leeuwenberg (1969)), in economy see Keuzenkamp and McAleer (1995), and in model selection and prediction see Vitányi and Li (2000). Here we give three applications of Kolmogorov complexity related to social sciences, explain the novel 'incompressibility method,' and conclude with an elementary proof of Gödel's celebrated result that mathematics is undecidable.

3.1 *Cognitive Distance*

For a function $f(x, y)$ to be a proper *distance measure* we want it to be a *metric*: it has non-negative real values; it is symmetrical, $f(x, y) = f(y, x)$; it satisfies the triangle inequality, $f(x, y) \leqslant f(x, z)+f(z, y)$; and $f(x, y) = 0$ iff $x = y$. Given two objects, say two pictures, how do we define an objective measure that would define their distance that is universal in the sense that it accounts for all cognitive similarities? Traditional distances do not work. For example, given a picture and its negative (i.e., exchange 0 and 1 in each pixel), Hamming distance and Euclidean distance both fail to recognize their similarity. Let us define a new distance $D(x, y)$ between two objects x and y as the length of the shortest program that converts them back and forth (Bennett et al. 1998). It turns out that, up to a logarithmic additive term,

$$D(x, y) = \max\{C(x|y), C(y|x)\} \qquad (2)$$

This distance D is a proper metric and it is *universal* in the sense that if two objects are 'close' under any distance out of a wide class of sensible and computable metrics, then they are also 'close' under d. For example, the $D(x, y)$ distance between two black-and-white pictures x and its negative y is a small constant.

3.2 *Phylogeny of Chain Letters* (*and Biological Evolution*)

Chain letters are an interesting social phenomenon that have reached billions of people. Such letters evolve, much like biological species (rather, their genomes). Given a set of genomes we want to determine the evolutionary history (phylogeny tree). Can we use the information distance $D(x, y)$? But then the difference in length of (especially complex) genomes implies a large distance while evolutionarily the genomes concerned can be very close (part of the genome was simply erased). We can divide $D(x, y)$ by some combination of the lengths of x and y, but this can be shown to be improper as well. As we have seen, $C(y) - C(y|x) = C(x) - C(x|y)$ within a logarithmic additive constant: it is the mutual information between x and y. But mutual information itself does not satisfy the triangle inequality and hence is not a metric and therefore clearly cannot be used to determine phylogeny. The solution is to determine closeness between each pair of genomes (or pairs of chain letters) x and y by taking the ratio of the information distance to the maximal complexity of the two:

$$d(x, y) = \frac{D(x, y)}{\max\{C(x), C(y)\}} \tag{3}$$

Note that $d(x, y)$ is always a sort of normalized dissimilarity coefficient that is at most 1. Moreover, it is proper metric. Let us look a little bit closer: suppose $C(y) \geqslant C(x)$. Then, up to logarithmic additive terms in both nominator and denominator we find (using Eqn. (2))

$$d(x, y) = \frac{C(y|x)}{C(y)} = 1 - \frac{C(y) - C(y|x)}{C(y)} \tag{4}$$

It turns out that $d(x, y)$ is universal (always gives the smallest distance) in a wide class of sensible and computable normalized dissimilarity coefficient metrics. It measures the percentage of shared information, which is a convenient way to measure English text or DNA sequence similarity. We have actually applied this measure (or rather, a less perfect close relative) to English texts. Using a compression program called *GenCompress* we heuristically approximate $C(x)$ and $C(x|y)$. With the caveat 'heuristic,' that is, without mathematical closeness-of-approximation guarantees, C. H. Bennett, M. Li and B. Ma (in an article to appear in *Scientific American*) took 33 chain letters—collected by Charles Bennett from 1980 to 1997—and approximated their pairwise distance $d(x, y)$. Then, we used standard phylogeny building programs from bioinformatics research to construct a tree of these chain letters. The resulting tree gives a perfect phylogeny for all notable features, in the sense that each notable feature is grouped together in the tree (so that the tree is parsimonious). This fundamental notion can be applied in many different areas. One of these concerns a major challenge in bioinformatics: to find good methods to compare genomes. Traditional approaches of computing the phylogeny use so-called 'multiple alignment.' They would not work here since chain letters contain swapped sentences and genomes contain translocated genes and noncoding regions. Using the chain letter method, a more serious application in Li et al. (2001) automatically builds correct phylogenies from complete mitochondrial genomes of mammals. We confirmed a biological conjecture that ferungulates—placental mammals that are not primates, including cats, cows, horses, whales—are closer to the primates—monkeys, humans—than to rodents.

3.3 *Inductive Reasoning*

Solomonoff (1964) argues that all inference problems can be cast in the form of extrapolation from an ordered sequence of binary symbols. A principle to enable us to extrapolate from an initial segment of a sequence to its continuation will either require some hypothesis about the source of the sequence or another method to do the extrapolation. Two popular and useful metaphysical principles for extrapolation are those of simplicity (Occam's razor, attributed to the thirteenth-century scholastic philosopher William of Ockham, but emphasized about 20 years before Ockham by John Duns Scotus), and indifference. The Principle of Simplicity asserts that the 'simplest' explanation is the most reliable. The Principle of Indifference asserts that in the absence of grounds enabling us to choose between explanations we should treat them as equally reliable. Roughly, the idea is to define the universal probability, $\mathbf{M}(x)$, as the probability that a program in a fixed universal programming language outputs a sequence starting with x when its input is supplied by tosses of a fair coin (see Kirchherr et al. 1997). Using this as a sort of 'universal prior probability' we then can formally do the extrapolation by Bayes's Rule. The probability that x will be followed by a 1 rather than by a 0 turns out to be

$$\frac{\mathbf{M}(x1)}{\mathbf{M}(x0) + \mathbf{M}(x1)}$$

It can be shown that $-\log \mathbf{M}(x) = C(x)$ up to an additive logarithmic term, which establishes that the distribution $\mathbf{M}(x)$ is a mathematical version of Occam's razor: low complexity xs have high probability ($x = 11\ldots1$ of every length n has complexity $C(x) \leqslant \log n + O(1)$ and hence universal probability $\mathbf{M}(x) \geqslant 1/n^c$ for some fixed constant c), and high complexity ys have low probability (if y is the outcome of n flips of a fair coin then for example with probability 0.9999 we have $C(y) \geqslant n - 10$ and therefore $\mathbf{M}(x) \leqslant 1/2^{n-10}$). This theory was further developed in Li and Vitányi (1993), Kirchherr et al. (1997) and Vitányi and Li (2000), and relates to more informal cognitive psychology work starting with Leeuwenberg (1969) and the applied statistical 'minimum description length (MDL)' model selection and prediction methods surveyed in Barron et al. (1998).

3.4 *Incompressibility Method*

Analyzing the performance of computer programs is very difficult. Analyzing the average case performance of computer programs is often more difficult since one has to consider all possible inputs and take the average. However, if we could find a typical input on which the program takes an average amount of time, then all we need to do is to find the performance of the computer program on this particular input. Then the analysis is easy. A Kolmogorov random input does exactly that: providing a typical input. Using this method, we were able to solve many otherwise difficult problems. Recent examples are average case analysis of Shellsort algorithm (Jiang et al. in press) and the average case of Heilbronn's triangle problem. A popular account about how to analyze the average-case bounds on Heilbronn's triangle problem can be found in Mackenzie (1999).

3.5 *Gödel's Incompleteness Result*

A new elementary proof by Kolmogorov complexity of K. Gödel's famous result showing the incompleteness of mathematics (not everything that is true can be proven) is due to Ya. Barzdin's and was later popularized by G. Chaitin, see Li and Vitányi (1993). A formal system (consisting of definitions, axioms, rules of inference) is consistent if no statement that can be expressed in the system can be proved to be both true and false in the system. A formal system is sound if only true statements can be proved to be true in the system. (Hence, a sound formal system is consistent.)

Let x be a finite binary string. We write 'x is random' if the shortest binary description of x with respect to the optimal specification method D_0 has length at least $|x|$. A simple counting argument shows that there are random xs of each length.

Fix any sound formal system F in which we can express statements like 'x is random.' Suppose F can be described in f bits—assume, for example, that this is the number of bits used in the exhaustive description of F in the first chapter of the textbook *Foundations of F*. We claim that for all but finitely many random strings x, the sentence 'x is random' is not provable in F. Assume the contrary. Then given F, we can start to exhaustively search for a proof that some string of length $n \gg f$ is random, and print it when we find such a string x. This procedure to print x of length n uses only $\log n + f$ bits of data, which is much less than n. But x is random by the proof and the fact that F is sound. Hence, F is not consistent, which is a contradiction. This shows that although most strings are random, it is impossible to effectively prove them random. In a way, this explains why the incompressibility method above is so successful. We can argue about a 'typical' individual element, which is difficult or impossible by other methods.

See also: Algorithms; Computational Approaches to Model Evaluation; High Performance Computing; Information Processing Architectures: Fundamental Issues; Information Theory; Mathematical Psychology; Model Testing and Selection, Theory of

Bibliography

Barron A R, Rissanen J, Yu B 1998 The minimum description length principle in coding and modelling. *IEEE Trans. Inform. Theory* **44**(6): 2743–60

Bennett C H, Gács P, Li M, Vitányi P, Zurek W 1998 Information distance. *IEEE Trans. Inform. Theory* **44**(4): 1407–23

Bennett C H, Li M, Ma B, in press Linking chain letters, *Scientific American*

Chaitin G J 1969 On the length of programs for computing finite binary sequences: Statistical considerations. *J. Assoc. Comput. Mach.* **16**: 145–59

Chater N 1996 Reconciling simplicity and likelihood principles in perceptual organization. *Psychological Review* **103**: 566–81

Chen X, Kwong S, Li M 1999 A compression algorithm for DNA sequences and its application in genome comparison. In GIW'99, Tokyo, Japan, Dec. 1999, and in *RECOMB'00*. Tokyo, Japan, April 2000

Jiang T, Li M, Vitányi P in press A lower bound on the average-case complexity of Shellsort. *J. Assoc. Comp. Machin.* **47**: 905–11

Keuzenkamp H A, McAleer M 1995 Simplicity, scientific inference and econometric modelling. *The Economic Journal* **105**: 1–21

Kirchherr W W, Li M, Vitányi P M B 1997 The miraculous universal distribution. *Mathematical Intelligencer* **19**(4): 7–15

Kolmogorov A N 1965 Three approaches to the quantitative definition of information. *Problems Information Transmission* **1**(1): 1–7

Koonin E V 1999 The emerging paradigm and open problems in comparative genomics. *Bioinformatics* **15**: 265–6

Leeuwenberg Q J 1969 Quantitative specification of information in sequential patterns. *Psychological Review* **76**: 216–20

Li M, Badger J, Chen X, Kwong S, Kearney P, Zhang H 2001 An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics* **17**(2): 149–54

Li M, Vitányi P 1993 *An Introduction to Kolmogorov Complexity and its Applications*, 1st edn. Springer Verlag, New York

Mackenzie D 1999 On a roll. *New Scientist* **164**: 44–7

Solomonoff R J 1964 A formal theory of inductive inference, Part 1 and Part 2. *Information Control* **7**: 1–22, 224–54

Turing A M 1936 On computable numbers with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society* **2**(42): 230–65

Vitányi P M B, Li M 2000 Minimum description length induction, Bayesianism, and Kolmogorov complexity. *IEEE Trans. Inform. Theory* **46**

Wooley J C 1999 Trends in computational biology: A summary based on a RECOMB plenary lecture, 1999. *Journal of Computational Biology* **6**(314): 459–74

M. Li and P. Vitányi

Algorithms

1. *Introduction*

Around the year 825 the Persian mathematician Abu Ja'far Mohammed ibn Mûsâ al-Khowârizm wrote a textbook entitled *Kitab al jabr w'al muqabala.* The term 'algorithm' is directly derived from the last part of the author's name. An algorithm is a mathematical recipe, formulated as a finite set of rules to be performed systematically, that has as outcome the solution to a well-formulated problem. In sequential algorithms these steps are ordered and should be performed one after the other. In parallel algorithms some of the rules are to be performed simultaneously. Algorithms can be graphically represented by flowcharts composed by arrows and boxes. The boxes contain the instructions and the arrows indicate transitions from one step to the next.

Algorithms were known much earlier than the eighth century. One of the most familiar, dating from ancient Greek times (c. 300 BC), is the procedure now referred to as Euclid's algorithm for finding the highest common factor of two natural numbers.

Algorithms often depend on subalgorithms or subroutines. For instance, the algorithm for obtaining the highest common factor of two numbers relies on the algorithm for finding the remainder of division for two natural numbers *a* and *b*. Dividing two numbers a and b is something we learn at school after having learnt the multiplication tables by heart. Usually we perform a division by reducing it to a sequence of multiplications and subtractions. Yet assume for the moment that to perform division we cannot rely on multiplication, nor can we express numbers in base ten. Both of these operations will require additional algorithms. We represent numbers in a very primitive form by simply writing a sequence of dots. Thus ●●●●●, for instance, represents the number 5.

The remainder algorithm works as follows (using $17 \div 5$ as an example):

(a) Write the sequence of dots for the dividend (17 in this case).
(b) Erase as many dots as correspond to the divisor (5 in this case).
(c) If the remaining number of dots is larger than or equal to the divisor go back to (a)
(d) If the remaining number of dots is smaller than the dividend print out this number
(e) STOP

The algorithm just described contains a loop. Observe that for any pair of numbers the algorithm produces the answer in a finite number of steps. Applied to our example of $17 \div 5$, the algorithm performs the following steps:

Current state	Operation
START	Write 17 dots
●●●●●●●●●●●●●●●●●	Erase 5 dots
●●●●●●●●●●●●	Erase 5 dots
●●●●●●●	Erase 5 dots
●●	Print 2 dots
STOP	

The computational algorithm for finding the remainder of a number when divided by another number can be used as a subroutine of the decision algorithm for the decidable problem 'Does *b* divide *a*?' (the answer is 'yes' if the remainder is zero). Repeated application of these algorithms produces the answer to the decidable question 'Is *a* a prime?' (the answer is 'no' if *a* is divisible by any smaller natural number besides 1).

An algorithm or machine is deterministic if at each step there is only one possible action it can perform. A nondeterministic algorithm or machine may make random choices of its next action at some steps. An algorithm is called a decision algorithm if it leads to a 'yes' or a 'no' result, whereas it is called a computational algorithm if it computes a solution to a given well-defined problem.

Despite the ancient origins of specific examples of algorithms, the precise formulation of the concept of a general algorithm dates only from the last century. The first rigorous definitions of this concept arose in the 1930s. The classical prototype algorithm is the Turing machine, defined by Alan Turing to tackle the *Entscheidungsproblem* or Decision Problem, posed by the German mathematician David Hilbert in 1900, at the Paris International Congress of Mathematicians. Hilbert's dream was to prove that the edifice of mathematics is a consistent set of propositions derived from a finite set of axioms, from which the truth of any well-formulated proposition can be established by a well-defined finite sequence of proof steps. The development and formalization of mathematics had led mathematicians to see it as the perfect, flawless science.

2. *Algorithms and the* Entscheidungsproblem

In 1931 the foundation of mathematics suffered its most crushing blow from a startling theorem proven by the Austrian logician Kurt Gödel. Gödel showed that any mathematical system powerful enough to represent arithmetic is incomplete in the sense that there must exist propositions that cannot be proven true or untrue in a finite sequence of steps. Such propositions are said to be undecidable within the given system. Turing had been motivated by Gödel's work to seek an algorithmic method of determining