Algorithmic information theory

Alexander Shen

December 26, 2014

Algorithmic information theory uses the notion of algorithm to measure the amount of information in a finite object. The corresponding definition was suggested in 1960s by Ray Solomonoff, Andrei Kolmogorov, Gregory Chaitin and others: the amount of information in a finite object, or its *complexity*, was defined as the minimal length of a program that generates this object.

Informally, the amount of information in a finite object can be described as the number of bits (zeros and ones) needed to encode this message. For example, there are 26 Latin letters, so one can encode a letter by a combination of five bits (there are $2^5 = 32$ bit strings of length 5, and $32 \geq 26$). On the other hand, it is not possible to encode Latin letters by 4-bit strings, because we do not have enough 4-bit strings ($2^4 = 16 < 32$). So we can say that each Latin letter carries between 4 and 5 bits of information. In general, if there are N possible messages and we agree in advance on the list of all possible messages, then each message can be encoded by $\lceil \log_2 N \rceil$ bits.

More complicated situation arises when we know in advance that some messages are more probable than others and want to minimize the expected number of bits per message. Then we can use shorter encodings for more popular messages, like in the Morse code where short sequences of dots and dashes are used for some frequent letters. This approach leads to the notion of Shannon entropy of a random variable (see the chapter about Shannon information theory).

In both cases we assume that the set of possible messages (and a probability distribution on it, for the case of Shannon entropy) is fixed in advance. So we cannot measure the amount of information in an individual object (such as DNA sequence or a novel) in this way. Indeed, it is not clear what set of "possible DNA sequences" should we consider as an ensemble from which our sequence is taken. If we choose the set of all DNA sequences that existed on earth, the amount of information would be at most few hundred bits, a ridiculously small quantity; moreover, this set increases over time. Similarly, it is absurd to say that *The Brothers Karamazov* novel contains at most 4 bits of information on the grounds that Dostoevsky (its author) has written at most 16 novels.

Algorithmic information theory overcomes this problem in the following way. The amount of information in an individual finite object, or its *Kolmogorov complexity*, is defined as *the minimal length of a program without input that generates this object*. In this definition programs and objects are binary strings. A programming language is specified by fixing an interpreter for it, i.e., an algorithm D such that $D(p)$ is the output generated by program p. We consider all algorithms whose inputs and outputs are binary strings, as interpreters. Then for a string x its *complexity with respect to D* is defined as $C_D(x) = \min\{l(p) \mid D(p) = x\}$ where $l(p)$ stands for the length of p, and the minimum of the empty set is $+\infty$.

The function C_D depends on the choice of D; better (more efficient) languages D may provide shorter programs and lower complexity than less efficient ones. One would like to find the optimal, i.e., most efficient, programming language in this sense. However, an obvious problem arises. It may happen that some D is very efficient for some class of objects, i.e., objects of this class have short programs, and at the same time D is less efficient for other objects. For each object x we can design a special programming language D_x where x has a very short program. For example, we may agree that a one-bit string 0 is a program for x: the interpreter D_x, seeing the input 0, interprets it as a command to produce x. So the complexity $C_D(x)$ is 1 if D_x is chosen as D. Moreover, having two arbitrary objects (strings) x and y, we may consider an interpreter D such that $D(0) = x$ and $D(1) = y$, thus making both complexities $C_D(x)$ and $C_D(y)$ equal to 1. However, we cannot assign short programs to many different objects, as there is not enough short programs for that. This argument shows that there is no interpreter that is better than every program of the form D_x.

So a weaker asymptotic notion of optimality is needed. We say that *D is better than D'* if $C_D(x) \le C_{D'}(x) + c$ for some c and for all x. The constant c in the definition means that we are ready to accept longer programs as soon as the overhead is bounded by some constant that depends only on the choice of D' and is independent of x. An interpreter is *optimal* if it is better than any other one. The *Kolmogorov–Solomonoff optimality theorem* says that optimal interpreters exist. We fix arbitrarily some optimal interpreter D. The value $C_D(x)$ is then called *Kolmogorov* complexity of x and denoted by $C(x)$. Using different optimal interpreters D_1 and D_2, we get different complexity functions C_{D_1} and C_{D_2}, but the optimality property guarantees that the difference is bounded: $|C_{D_1}(x) - C_{D_2}(x)| \le c$ for some constant c (depending on D_1 and D_2 but not on x) and for all x. So Kolmogorov complexity is defined up to a bounded additive term.

The Kolmogorov complexity of a string x is bounded by the length of x up to some bounded additive term: all n-bit strings have complexity at most $n+c$, where c is a constant not depending on n. Indeed, every n-bit string x can be generated by a program "output $XX\ldots X$", where $XX\ldots X$ are the bits of x, and this program has length about n. So the complexity of a n-bit string cannot be much bigger than n (asymptotically, as n goes to infinity). But it may be much smaller than n; for example, the complexity of the string 0^n (n zeros) does not exceed $\log n + c$ for some c and for all n. Indeed, 0^n can also be generated by a shorter program "output $NN\ldots N$ zeros" where $NN\ldots N$ is n written in binary, and the length of this shorter program is about $\log n$. A counting argument shows that most strings of length n are *incompressible*, i.e., have complexity close to n. Indeed, compressible strings have programs whose length is significantly smaller than n. Using some $k<n$ as a threshold, we note that at most 2^k strings may have programs shorter than k: different strings have different programs, and the total number of all possible programs shorter than k is bounded by a total number of strings of lengths $0,1,2,\ldots,k-1$, i.e., is bounded by

$$1+2+2^2+\ldots+2^{k-1}=2^k-1<2^k.$$

We see that if k is significantly smaller than n, then the strings of complexity less than k form only a very small fraction ($1/2^{n-k}$) of all n-bit strings.

Tossing a fair coin n times and recording heads and tails, we get a n-bit string that is incompressible with high probability, i.e., with probability close to 1, according to the uniform distribution on n-bit strings usually associated with "fair coin tossing". So incompressible strings are also called *random*. Note, however, that there is no strict boundary between compressible (non-random) strings and incompressible (random) ones; the difference between the length and the complexity, called sometimes the *randomness deficiency*, is small for random strings and large for non-random ones, but there is no specific threshold that separates one collection from the other.

The notion of Kolmogorov complexity has some properties that are intuitively expected for a measure of information. Here is one example of such a property. Let A be some algorithm whose inputs and outputs are strings. One can easily show that the complexity of the string $A(x)$ may exceed the complexity of x at most by some constant c that depends on A but not on x. Interpreting the Kolmogorov complexity of a string as the "amount of information" in this string measured in bits, we may say that the algorithmic transformation of a string cannot create new information and increases the complexity only by some constant (depending on how much information was in the algorithm itself). This result allows us to define Kolmogorov complexity of arbitrary finite objects (like graphs, rational numbers, logical formulas, etc.) as the complexity of their binary encoding. The choice of a computable encoding changes the complexity defined in this way by at most an additive constant.

In saying that Kolmogorov complexity measures the amount of information in a given string, we do not mean that one can actually perform such a measurement. The exact value of complexity depends on the choice of an optimal interpreter, so the question like "which of the strings 000100010001001 and 10010000000 has higher complexity" has no meaning, as the answer depends on the choice of the interpreter. Also one can prove that complexity function is non-computable: there is no algorithm that given x computes $C(x)$. Moreover, the complexity function has no non-trivial computable lower bounds; the proof imitates *Berry's paradox* about *the minimal natural number that cannot be described by twelve English words* (this description has exactly twelve words). Using this argument, Chaitin provided a proof of the Gödel incompleteness theorem based on Kolmogorov complexity; he showed that for large enough M one cannot prove any statement of the form "the complexity of x exceeds M" while these statements are true for all but finitely many x.

Kolmogorov complexity is sometimes called *descriptional complexity*. This should be distinguished from *computational complexity* where one studies the computational resources (processor steps, memory cells) needed to solve some problem. A string has small Kolmogorov complexity if there exists a short program that outputs it, even if this program needs a lot of time.

Information systems often store information in compressed form. A compressor program uses the regularities in the data, e.g., in some computer file, to compress this file and get a shorter one. This shorter file still contains all the information in the original file: one can restore the original file by applying a decompressor program to the compressed version. The size of the compressed version of a file is an upper bound for its Kolmogorov complexity (again up to some constant); such a bound depends on a specific method of compression and decompression used. Kolmogorov complexity, therefore, puts a limit to the possible compression of a file; no compressor can significantly compress a file whose Kolmogorov complexity is close to its length (such as a file that is a record of fair coin tossing).

When we say that Kolmogorov complexity measures the "amount of information" in a string, we do not distinguish between "useful" and "useless" information: a random noise has high Kolmogorov complexity but contains no "information" in the intuitive sense. Still we can define the amount of information in x about y for two binary strings x and y as the difference between the complexity of y and conditional complexity of y given x, the minimal length of a program that transforms x to y. To define the notion of conditional complexity in a more formal way, we consider algorithms $D(p,x)$ with two arguments as *conditional decompressors*. The first argument p is considered as a program, and the second argument x is considered as a condition. Then the function

$$C_D(y\,|\,x) = \min\{l(p) \mid D(p,x) = y\}$$

is defined. Again an optimal $D(p,x)$ exists that makes the function C_D minimal up to an additive constant. Some optimal $D(p,x)$ is fixed, and the corresponding function $C_D(y\,|\,x)$ is called *conditional complexity of y given x*.

Then the *amount of information in x about y* is defined as $C(y) - C(y\,|\,x)$. As shown by Kolmogorov and Leonid Levin, the notion of information defined in this way is almost symmetric: the amount of information in x about y and the amount of information in y about x differ at most by $c\log n$ for some c and for arbitrary n and arbitrary n-bit strings x and y. Both values are close to the *mutual information between x and y*, defined as $C(x) + C(y) - C(x,y)$. Here $C(x,y)$ stands for the complexity of a pair (x,y), i.e., the complexity of some its computable encoding.

Whereas the Kolmogorov complexity of a string x measures how many bits are needed to specify this string, there is a related notion, called *a priori probability* of a string x that measures how likely x appears as an output of a random process. Consider a randomized algorithm M without input that uses fair coin tosses, outputs some binary string (depending on the outcome of the coin tosses), and terminates. The algorithm M may also work infinitely long producing no output. By $p_M(x)$ we denote the probability of the event "the binary string x appears as the output of M". In this way for every M we get a real-valued function p_M defined on binary strings; these functions are called *lower semicomputable semimeasures*. Among all these function there exists a maximal one: there is some algorithm M such that for every algorithm M' the inequality $p_M(x) \geq \varepsilon p_{M'}(x)$ is true for some $\varepsilon > 0$ and for all x. We fix one of these maximal functions p_M and call it *discrete a priori probability of x*. As shown by Levin, it is closely related to Kolmogorov complexity: $\log_2(1/p_M(x))$ differs from $C(x)$ at most by $c\log l(x)$ for some constant c and for all x. Moreover, Levin suggested a different definition of complexity, called *prefix complexity*, that makes the difference between $\log_2(1/p_M(x))$ and the complexity of x bounded. Later this connection between a priori probability and complexity was rediscovered by Chaitin.

The algorithmic information theory may be considered as an extension of Shannon information theory to individual objects. The Shannon entropy provides an upper bound for Kolmogorov complexity in the following sense: the complexity of a N-bit string x does not exceed $NH + O(\log N)$, where H is the Shannon entropy of a random variable with two values whose probabilities p_0 and p_1 are frequencies of zeros and ones in x. Intuitively, Shannon entropy takes into account the statistical regularities while Kolmogorov complexity considers all algorithmically discoverable regularities, including the statistical ones, so Shannon entropy provides an upper bound for Kolmogorov complexity. One can show also that if a finite object is generated by a random process, then with probability close to 1 the Kolmogorov complexity of this object is close to the entropy of the random process.

Another, more recent result that relates algorithmic information theory and classical information theory (by Andrei Romashchenko): the linear inequalities that are true for Shannon entropies of tuples of variables, and for Kolmogorov complexities of tuples of strings (with logarithmic precision in the latter case) are the same.

Switching from finite binary strings to infinite sequences of zeros and ones, one can draw a sharp dividing line between "random" and "non-random" sequences. The first attempts to provide such a definition were made in the beginning of the 20th century by Richard von Mises who defined a *Kollektiv* as a sequence where zeros and ones appear with some limit frequencies that remains the same for every subsequence selected by some admissible rule. Mises did not give a precise definition; later Abraham Wald, Jean Ville, Alonzo Church, Donald Loveland, Kolmogorov and others studied different mathematical notions of randomness defined along these lines. It turned out that this notion is too broad; in 1960s Per Martin-Löf suggested a stricter definition based on the ideas of constructive measure theory. A set X of binary sequences is *effectively null* if for every n one can effectively generate a sequence of intervals that covers X and has measure less than 2^{-n}. Martin-Löf proved that there exists a maximal effectively null set containing all others; *Martin-Löf random* sequences are sequences that do not belong to this maximal set. This notion is related to the notion of incompressibility: Claus-Peter Schnorr and Levin proved that a binary sequence is Martin-Löf random if and only if its finite prefixes are incompressible (have bounded randomness deficiency, where randomness deficiency is defined using special version of complexity called *monotone* complexity; prefix complexity can also be used). Martin-Löf's definition of randomness guarantees that most laws of probability theory hold for random sequences. For example, the *strong law of large numbers* says that with probability 1 a sequence of zeros and ones obtained by tossing a fair coin is *normal*, i.e., all combinations of n zeros and ones appear in the sequence with the same limit frequency 2^{-n}. The constructive version of this law guarantees that *every* Martin-Löf random sequence is normal. However, not every normal sequence is random: the *Champernown sequence* 011011100101110…, obtained by concatenation of all integers in binary (0, 1, 10, 11, 100, 101, 110,…), is normal but computable, and no computable sequence is Martin-Löf random.

There are interesting connections between computability (recursion) theory and algorithmic information theory, see [8, 9].

The notion of complexity is often used while discussing the methodology of natural sciences. The famous Occam's razor asks for the simplest possible explanation of some observations (sometimes one speaks also about the 'economy of thought'), but does not say how we measure the 'simplicity'. With all disclaimers above (Kolmogov complexity is not uniquely defined and is not computable), we can use the algorithmic information theory to make this notion of 'simplicity' more concrete. A more specific version of this approach is called the *minimal description principle*. Having some experimental data, presented as a binary string x, we look for a statistical model that treats this x as a typical example taken from some class S of strings. More formally, we consider a finite set of strings S that contains x, as a model for x. There could be several models for the same x; which is better? According to the minimal description length principle, a good model, called *sufficient statistic*, should have $C(S) + \log_2 \#S$ close to $C(x)$, where $\#S$ stands for the cardinality of S, and $C(S)$ is the complexity of S (defined as the complexity of the binary encoding of S; recall that S is a finite set of binary strings, so it can be encoded by a binary string). Each element of S has a two-part description. First we specify S using $C(S)$ bits. Then we specify the ordinal number of x in S, i.e., we specify that x is the n-th element of S in some fixed natural ordering. This second part n uses $\log_2 \#S$ bits. In total we use about $C(S) + \log_2 \#S$ bits for this two-part description, so $C(x)$ cannot exceed significantly $C(S) + \log_2 \#S$. The minimal description length principle says that for a good model these two values are close to each other. In this way we may hope to separate the regularities in x that force it to be in S, and random noise that determines which specific element of S was chosen. If there are several sufficient statistics, the *minimal sufficient statistic* where $C(S)$ is as small as possible, should be preferred.

The notions of complexity and randomness are important for the foundations of probability theory. A well-known paradox: seeing thousand tails in a row while tossing a coin, we reject the hypothesis of a fair coin on the grounds that the probability of such an event is astronomically small, only 2^{-1000}. However, if the other day some other sequence y of 1000 heads and tails appears, the probability to see this sequence y is the same 2^{-1000}, yet we do not consider it as a reason to reject the fair coin hypothesis. What is the difference between the sequence x of 1000 tails and the sequence y? One can say that x was in our mind before the experiment while y was not; however, if a sequence with 1000 first binary digits of π appears, we also will suspect some cheating trick even if we never thought about this possibility before the experiment. Using the notion of complexity, we may explain the difference saying that x is simple while the complexity of y is close to its length.

There are many other philosophical questions related to the notion of complexity. For example, there is an experimental observation: tossing a fair coin many times and applying standard compression software to the resulting sequence, we never achieve a significant compression. Is this observation derivable from the laws of physics, or it is an additional law of nature? Does it happens because the coin tossing results have high Kolmogorov complexity, or just because our compressors are not clever enough? Can Kolmogorov complexity be applied somehow to the foundations of thermodynamics—in particular, can it be used to define entropy of a specific state of a dynamical system?

Though the notion of Kolmogorov complexity is purely theoretical, the ideas from algorithmic information theory can give insights for practical applications. Rudi Cilibrasi and Paul Vitanyi suggested to use compressed size as a first approximation to complexity, and defined a version of information distance: the distance between x and y is small if the concatenated string xy, being compressed, gives something much shorter than the concatenation of compressed versions of x and y. They applied this distance to the hierarchical clustering problems. They also tried another approach that used the number of appearences in a search engine as a way to estimate a priori probability.

Literature. Original papers: In [1] the notion of Kollektiv is introduced; [2, 3, 4] contain the definition of complexity (independently discovered); [5] describes the proof of Gödel incompleteness theorem using complexity; in [6] the definition of Martin-Löf random sequences is given. Textbooks and surveys: [7] covers a lot of material related to Kolmogorov complexity and algorithmic randomness, with historical account and references; [8, 9] also cover more recent results relating algorithmic information theory and recursion (computability) theory; [10] is a textbook that also contains some more recent results not covered by other books; [11] is a concise introduction to the subject (lecture notes of a course), and [12] is a survey describing philosophical aspects of algorithmic information theory.

References

[1] von Mises, Richard. Grundlagen der Wahrscheinlichkeitsrechnung, *Mathematische Zeitschrift*, **5**, 52–99 (1919). Reprinted in *Selected papers of Richard von Mises*. Volume 2. Probability and Statistics, General. American Mathematical Society, Providence, RI, 1964.

[2] Solomonoff, Ray. A formal theory of inductive inference, *Information and Control*, **7**(1), 1–22 (1964), especially Section 3.1.2.

[3] Kolmogorov, Andrei N. Three approaches to the quantitative definition of information. Russian original published in *Problemy peredachi informatsii* [Problems of Information Transmission], **1**(1), 3–11 (1965). Translation is reprinted in *International Journal of Computer Mathematics*, **2**, 157–168 (1968).

[4] Chaitin, Gregory. On the length of programs for computing finite binary sequences: statistical considerations. *Journal of the ACM*, **16**(1), 145–159 (1969), especially Section 9.

[5] Chaitin, Gregory. Computational complexity and Gödel's incompleteness theorem, *ACM SIGACT News*, **9** (April 1971), 11–12.

[6] Martin-Löf, Per. The definition of random sequences, *Information and Control*, **9**, 602–619 (1966).

[7] Li, Ming and Vitanyi, Paul M.B. *An Introduction to Kolmoghorov complexity*, 3rd edition, Springer, 2008. XXIV, 792 p.

[8] Downey, Rodney G. and Hirschfeldt, Denis R., *Algorithmic Randomness and Complexity*, Springer, 2010. XXIII, 855 p.

[9] Nies, André. *Computability and Randomness*, Oxford University Press, 2012, 456 p.

[10] Vereshchagin, Nikolay K., Uspensky, Vladimir A. and Shen, Alexander, *Kolmorogov complexity and algorithmic randomness*, Moscow, MCCME Publishers, 2012. [In Russian; for draft English translation see `www.lirmm.fr/~ashen/kolmbook-eng.pdf`]

[11] Shen, Alexander, *Algorithmic information theory and Kolmogov complexity*, Technical Report TR2000-034, Uppsala University, 2000. Available at `www.it.uu.se/research/publications/reports/2000-034/`.

[12] Shen, Alexander, Algorithmic information theory and foundations of probability, *Reachability problmes, LNCS 5797*, Springer, 2009, 26–34. See also `arxiv.org/abs/0906.4411`.