Algorithmic Information Theory

[a brief non-technical guide to the field]

Marcus Hutter

RSISE @ ANU and SML @ NICTA
Canberra, ACT, 0200, Australia
marcus@hutter1.net   www.hutter1.net

March 2007

Abstract

This article is a brief guide to the field of algorithmic information theory (AIT), its underlying philosophy, and the most important concepts. AIT arises by mixing information theory and computation theory to obtain an objective and absolute notion of information in an individual object, and in so doing gives rise to an objective and robust notion of randomness of individual objects. This is in contrast to classical information theory that is based on random variables and communication, and has no bearing on information and randomness of individual objects. After a brief overview, the major subfields, applications, history, and a map of the field are presented.

Contents

1 Overview

Algorithmic Information Theory (AIT) is a the information theory of individual objects, using computer science, and concerns itself with the relationship between computation, information, and randomness.

The information content or complexity of an object can be measured by the length of its shortest description. For instance the string "0101010101010101010101010101010101010101010101010101010101010101" has the short description "32 repetitions of '01"', while "1100100001100001110111101110110011111010010000100101011110010110" presumable has no simple description other than writing down the string itself. More formally, the Algorithmic "Kolmogorov" Complexity (AC) of a string x is defined as the length of the shortest program that computes or outputs x, where the program is run on some fixed universal computer.

A closely related notion is the probability that a universal computer outputs some string x when fed with a program chosen at random. This Algorithmic "Solomonoff" Probability (AP) is key in addressing the old philosophical problem of induction in a formal way.

The major drawback of AC and AP are their incomputability. Time-bounded "Levin" complexity penalizes a slow program by adding the logarithm of its running time to its length. This leads to computable variants of AC and AP, and Universal "Levin" Search (US) that solves all inversion problems in optimal (apart from some huge multiplicative constant) time.

AC and AP also allow a formal and rigorous definition of randomness of individual strings that does not depend on physical or philosophical intuitions about nondeterminism or likelihood. Roughly, a string is Algorithmically "Martin-Loef" Random (AR) if it is incompressible in the sense that its algorithmic complexity is equal to its length.

AC, AP, US, and AR are the core subdisciplines of AIT, but AIT spans into many other areas. It serves as the foundation of the Minimum Description Length (MDL) principle, can simplify proofs in computational complexity theory, has been used to define a universal similarity metric between objects, solves the Maxwell demon problem, and many others.

2 Algorithmic "Kolmogorov" Complexity (AC)

Algorithmic complexity formalizes the concept of simplicity and complexity. Intuitively, a string is simple if it can be described in a few words, like "the string of one million ones", and is complex if there is no such short description, like for a random string whose shortest description is specifying it bit by bit. Typically one is only interested in descriptions or *codes* that are effective in the sense that decoders are *algorithms* on some computer. The universal Turing machine U is the standard abstract model of a general-purpose computer in theoretical computer science. We say that program p is a description of string x if p run on U outputs x, and write $U(p) = x$. The length of the shortest description is denoted by

$$K(x) := \min_p \{\ell(p) : U(p) = x\}$$

where $\ell(p)$ is the length of p measured in bits. One can show that this definition is nearly independent of the choice of U in the sense that $K(x)$ changes by at most an additive constant independent of x. The statement and proof of this invariance theorem in [Sol64, Kol65, Cha69] is often regarded as the birth of algorithmic information theory. This can be termed Kolmogorov's Thesis: the intuitive notion of 'shortest effective code' in its widest sense is captured by the formal notion of Kolmogorov complexity, and no formal mechanism can yield an essentially shorter code. Note that the shortest code is one for which there is a general decompressor: the Kolmogorov complexity establishes the ultimate limits to how short a file can be compressed by a general purpose compressor.

There are many variants, mainly for technical reasons: The historically first "plain" complexity, the now more important "prefix" complexity, and many others. Most of them coincide within an additive term logarithmic in the length of the string.

In this article we use K for the prefix complexity variant. A prefix Turing machine has a separate input tape which it reads from left-to-right without backing up, a separate worktape on which the computation takes place, and a separate output tape on which the output is written. We define a halting program as the initial segment of the input that is scanned at the time when the machine halts, and the output is the string that has been written to the separate output tape at that time. The conditional prefix complexity

$$K(x|y) := \min_p \{\ell(p) : U(y,p) = x\}$$

is the length of the shortest binary program $p \in \{0,1\}^*$ on a universal prefix Turing machine U with output x and input y [LV97]. For non-string objects (like numbers n, pairs of strings (x,y), or computable functions f) one can specify some default coding $\langle \cdot \rangle$ and define $K(\mathit{object}) := K(\langle \mathit{object} \rangle)$. The most important properties are:

- that K is approximable from above in the limit but not computable,
- the upper bounds $K(x|\ell(x)) \leq \ell(x)$ and $K(n) \leq \log n + 2\log\log n$,
- Kraft's inequality implies $\sum_x 2^{-K(x)} \leq 1$,
- the lower bound $K(x) \geq \ell(x)$ for "most" x and $K(x) \to \infty$ for $\ell(x) \to \infty$,
- extra information bounds $K(x|y) \leq K(x) \leq K(x,y)$,
- subadditivity $K(xy) \leq K(x,y) \leq K(y) + K(x|y)$,
- symmetry of information $K(x,y) = K(x|y, K(y)) + K(y) = K(y,x)$,
- information non-increase $K(f(x)) \leq K(x) + K(f)$ for computable functions f,
- and coding relative to a probability distribution (MDL) $K(x) \leq -\log P(x) + K(P)$ for computable probability distributions P,

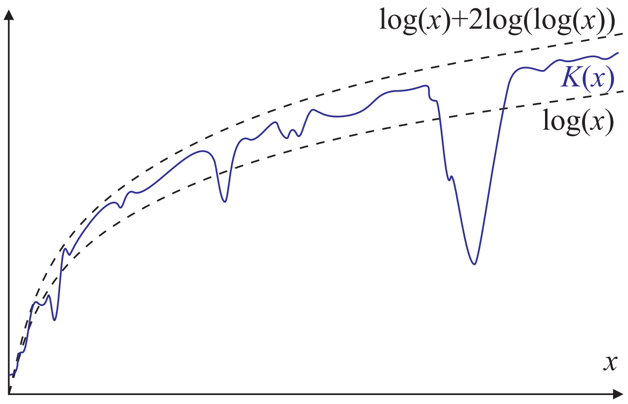

Figure 1: *Schematic graph of prefix Kolmogorov complexity $K(x)$ with string x interpreted as integer. $K(x) \geq \log x$ for 'most' x and $K(x) \leq \log x + 2\log\log x + c$ for all x for suitable constant c.*

where all (in)equalities hold within an additive constant. Furthermore, it shares many properties with Shannon's entropy (information measure), but K has many advantages. The properties above allow us to draw a schematic graph of K as depicted in Figure 1.

3 Algorithmic "Solomonoff" Probability (AP)

Solomonoff (1964) considered the probability that a universal computer outputs some string when fed with a program chosen at random. This Algorithmic "Solomonoff" Probability (AP) is key in addressing the old philosophical problem of induction in a formal way. It is based on

- Occam's razor (choose the simplest model consistent with the data),
- Epicurus' principle of multiple explanations (keep all explanations consistent with the data),
- Bayes's Rule (transform the a priori distribution to a posterior distribution according to the evidence, experimentally obtained data),
- (universal) Turing machines (to compute, quantify and assign codes to all quantities of interest), and
- algorithmic complexity (to define what simplicity/complexity means).

Occam's razor (appropriately interpreted and in compromise with Epicurus' principle of indifference) tells us to assign high/low a priori plausibility to simple/complex strings x. Using K as the complexity measure, one could choose any monotone decreasing function of K, e.g. $2^{-K(x)}$. The precise definition of Algorithmic "Solomonoff" Probability (AP), also called universal a priori probability, $M(x)$ is the probability that the output of a (so-called monotone) universal

Turing machine U starts with x when provided with fair coin flips on the input tape. Formally, M can be defined as

$$M(x) \ := \sum_{p\,:\,U(p)=x*} 2^{-\ell(p)}$$

where the sum is over all (so-called minimal, not necessarily halting, denoted by *) programs p for which U outputs a string starting with x. Since the shortest programs p dominate the sum, $M(x)$ is roughly $2^{-K(x)}$.

M has similar remarkable properties as K. Additionally, the predictive distribution $M(x_{n+1}|x_1...x_n) := M(x_1...x_{n+1})/M(x_1...x_n)$ converges rapidly to 1 on (hence predicts) any computable sequence $x_1x_2x_3$.... It can also be shown that M leads to excellent predictions and decisions in general stochastic environments. If married with sequential decision theory, it leads to an optimal reinforcement learning agent embedded in an arbitrary unknown environment [Hut05], and a formal definition and test of intelligence.

A formally related quantity is the probability that U halts when provided with fair coin flips on the input tape (i.e. that a random computer program will eventually halt). This halting probability, also known as Chaitin's constant Ω, or 'the number of wisdom' has numerous remarkable mathematical properties, and can be used for instance to quantify Goedel's Incompleteness Theorem.

4 Universal "Levin" Search (US)

Consider a problem to solve for which we have two potential algorithms A and B, for instance breadth versus depth first search in a finite (game) tree. Much has been written about which algorithm is better under which circumstances. Consider the following alternative very simple solution to the problem: A meta-algorithm US runs A and B in parallel and waits for the first algorithm to halt with the answer. Since US emulates A and B with half-speed, the running time of US is the minimum of $2\times$time(A) and $2\times$time(B), i.e. US is as fast as the faster of the two, apart from a factor of 2. Small factors like 2 are often minor compared to potentially much larger difference in running time of A and B.

Universal "Levin" Search (US) extends this idea from two algorithms to *all* algorithms. First, since there are infinitely many algorithms, computation time has to be assigned non-uniformly. The optimal way is that US devotes a time fraction of $2^{-\ell(p)}$ to each (prefix) program p. Second, since not all programs solve the problem (some never halt, some just print "Hello World", etc.) US has to verify whether the output is really a solution, and if not discard it and continue.

How does this fit into AIT? A problem of AC K is its incomputability. Time-bounded "Levin" complexity penalizes a slow program by adding the logarithm of its running time to its length:

$$Kt(x) \ = \ \min_p\{\ell(p) + \log(\text{time}(p)) : U(p) = x\}$$

It is easy to see that $Kt(x)$ is just the logarithm of the running time (without verification) of US, and is therefore computable.

While universal search is nice in theory, it is not applicable in this form due to huge hidden multiplicative constants in the running time. Another restriction is that verification needs to be fast. Hutter [Hut05] developed a more general asymptotically fastest algorithm, which removes the multiplicative constant and necessity of verification, unfortunately at the expense of an even larger additive constant. Schmidhuber [Sch06] developed the first practical variants of US by carefully choosing the programming language (U), allocating time in US adaptively, designing training sequences of increasing complexity, reusing subroutines from earlier simpler problems, and various other "tricks". He also defined the Speed Prior, which is to Kt what AP is to AC.

5 Algorithmic "Martin-Loef" Randomness (AR)

The mathematical formalization of the concept of probability or chance has a long intertwined history. The (now) standard axioms of probability, learned by all students, are due to Kolmogorov (1933).

While mathematically convincing, the semantics is far from clear. Frequentists interpret probabilities as limits of observed relatives frequencies, objectivists think of them as real aspects of the world, subjectivists regard them as one's degree of belief (often elicited from betting ratios), while Cournot only assigns meaning to events of high probability, namely as happening for sure in our world.

None of these approaches answers the question of whether some *specific individual* object or observation, like the binary strings above, is random. Kolmogorov's axioms do not allow one to ask such questions.

Von Mises (1919), with refinements to his approach by Wald (1937), and Church (1940) attempted to formalize the intuitive notion of one string looking more random than another (see the example in the introduction) with partial success. For instance, if the relative frequency of 1s in an infinite sequence does not converge to 1/2 it is clearly non-random, but the reverse is not true: For instance "0101010101..." is not random, since the pair "01" occurs too often. Pseudo-random sequences, like the digits of π, cause the most difficulties. Unfortunately no sequence can satisfy "all" randomness tests. The Mises-Wald-Church approach seemed satisfactory untill Ville (1939) showed that some sequences are random according to their definition and yet lack certain properties that are universally agreed to be satisfied by random sequences. For example, the relative frequency of '1's in increasingly long initial segments should infinitely often switch from above 1/2 to below 1/2 and vice versa.

Martin-Loef (1966), rather than give a definition and check whether it satisfied all requirements, took the approach to formalize the notion of all effectively testable requirements in the form of tests for randomness. The tests are constructive (namely all and only lower semi-computable) ones, which are typically all one ever cares about. Since the tests are constructed from Turing machines, they can be effectively enumerated according to the effective enumeration of the Turing machines they derive from. Since the set of sequences satisfying a test (having the randomness property the test verifies) has measure one, and there are only countably many tests, the set of sequences satisfying "all" such tests also has measure one. These are the ones called Algorithmic Random—Algorithmically "Martin-Loef" Random (AR). The theory is developed for both finite strings and infinite sequences. In the latter case the notion of test is more complicated and we speak of sequential tests.

For infinite sequences one can show that these are exactly the sequences which are incompressible in the sense that the algorithmic prefix complexity of every initial segment is at least equal to their length. More precisely, the infinite sequence

$$x_1x_2x_3... \text{ is AR} \quad \Longleftrightarrow \quad K(x_1...x_n) \geq n \text{ for all suff. large } n$$

an important result due to G.J. Chaitin and C. Schnorr. This notion makes intuitive sense: A string can be compressed "iff" there are some regularities in the string "iff" the string is non-random.

- ML-random sequences cannot be effectively constructed. Yet we can give a natural example: The halting probability, Ω is a real number between 0 and 1, and the sequence of bits in its binary expansion is an infinite ML-random sequence.
- Randomness of other objects than strings and sequences can also be defined.
- Coupling the theory of AR with recursion theory (Downey and Hirschfeldt 2007), we find a hierarchy of notions of randomness, at least if we leave the realm of computability according to Turing. Many variants can be obtained depending on the precise definition of "constructive". In particular "relative randomness" based on (halting) oracle machines leads to a rich field connected to recursion theory.
- Finally, the crude binary separation of random versus non-random strings can be refined, roughly by considering strings with $K(x_1...x_n) = \alpha n$ for some $0 < \alpha < 1$. If strings are interpreted as (the expansion of) real numbers, this leads to the notion of constructive or effective Hausdorff (fractal) dimension.

6 Applications of AIT

Despite the incomputability of its core concepts, AIT has many, often unexpected, applications.

Philosophy. AIT helps to tackle many philosophical problems in the sense that it allows one to formalize and quantify many intuitive but vague concepts of great importance as we have seen above, and hence allows one to talk about them in a meaningful and rigorous way, thus leading to a deeper understanding than without AIT.

Most importantly, AC formalizes and quantifies the concepts of simplicity and complexity in an essentially unique way. A core scientific paradigm is Occam's razor, usually interpreted as "among two models that describe the data equally well, the simpler one should be preferred." Using AC to quantify "simple" allowed Solomonoff and others to develop their universal theories of induction and action, in the field of artificial intelligence.

AIT is also useful in the foundations of thermodynamic and its second theorem about entropy increase, and in particular for solving the problem of Maxwell's demon.

Practice. By (often crudely) approximating the "ideal" concepts, AIT has been applied to various problems of practical interest, e.g. in linguistics and genetics. The principle idea is to replace the universal Turing machine U by more limited "Turing" machines, often adapted to the problem at hand. The major problem is that the approximation accuracy is hard to assess and most theorems in AIT break down.

The universal similarity metric by Vitanyi and others is probably the greatest practical success of AIT: A reasonable definition for the similarity between two objects is how difficult it is to transform them into each other. More formally one could define the similarity between strings x and y as the length of the shortest program that computes x from y (which is $K(x|y)$). Symmetrization and normalization leads to the universal similarity metric. Finally, approximating K by standard compressors like Lempel-Ziv (zip) or bzip(2) leads to the normalized compression distance, which has been used to fully automatically reconstruct language and phytogenetic trees, and many other clustering problems.

See Applications of AIT for details and references.

Science. In science itself, AIT can constructivize other fields: For instance, statements in Shannon information theory and classical probability theory necessarily only hold in expectation or with high probability. Theorems are typically of the form "there exists a set of measure X for which Y holds", i.e. they are useful for (large) samples. AR on the other hand can construct high-probability sets, and results hold for individual observations/strings. Hausdorff dimension and real numbers also have constructive counterparts.

Naturally, AIT concepts have also been exploited in theoretical computer science itself: AIT, via the incompressibility method, has resolved many open problems in computational complexity theory and mathematics, simplified many proofs, and is important in understanding (dissipationless) reversible computing. It has found applications in Statistics, Cognitive Sciences, Biology, Physics, and Economics.

AIT can also serve as an umbrella theory for other more practical fields, e.g., in machine learning, the Minimum Description Length (MDL) principle can be regarded as a downscaled practical version of AC.

7 History, References, Notation, Nomenclature

Andrey Kolmogorov [Kol65] suggested to define the information content of an object as the length of the shortest program computing a representation of it. Ray Solomonoff [Sol64] invented the closely related universal a priori probability distribution and used it for time series forecasting. Together with Gregory Chaitin [Cha69], this initiated the field of algorithmic information theory in the 1960s. Leonid Levin and others significantly contributed to the field in the 1970s (see e.g. [ZL70]). In particular the prefix complexity and time-bounded complexity are (mainly) due to him.

Li and Vitanyi [LV97] is the standard AIT textbook. The book by Calude [Cal02] focusses on AC and AR, Hutter [Hut05] on AP and US, and Downey and Hirschfeldt [DH07] on AR. The AIT website at http://www.hutter1.net/ait.htm contains further references, a list of active researchers, a mailing list, a list of AIT events, and more.

There is still no generally agreed upon notation and nomenclature in the field. One reason is that researchers of different background (mathematicians, logicians, and computer scientists) moved into this field. Another is that many definitions are named after their inventors, but if there are many inventors or one definition is a minor variant of another, things become difficult. This article uses descriptive naming with contributors in quotation marks.

Not even the name of the whole field is generally agreed upon. *Algorithmic Information Theory*, coined by Gregory Chaitin, seems most appropriate, since it is descriptive and impersonal, but the field is also often referred to by the more narrow and personal term *Kolmogorov complexity*.

8 Map of the Field

The AIT field may be subdivided into about 4 separate subfields: AC, AP, US, and AR. The fifth item below refers to applications.

- Algorithmic "Kolmogorov" Complexity (AC)
  - Philosophical considerations
  - Properties of AC
  - Plain (Kolmogorov) complexity
  - Prefix complexity
  - Resource bounded complexity
  - Other complexity variants
- Algorithmic "Solomonoff" Probability (AP)
  - Occam's razor and Epicurus' principle
  - Discrete algorithmic probability

  - Continuous algorithmic probability = a priori semimeasure
  - Universal sequence prediction
  - The halting probability = Chaitin's Omega = The number of Wisdom

- Universal "Levin" Search (US)

  - Levin search
  - Levin complexity and speed prior
  - Adaptive Levin search
  - Fastest algorithms for general problems
  - Optimal ordered problem solver
  - Goedel machines

- Algorithmic "Martin-Loef" Randomness (AR) / Recursion Theory

  - Recursion theory
  - Effective real numbers
  - Randomness of reals
  - van Mises-Wald-Church randomness
  - Martin-Loef randomness
  - More randomness concepts and relative randomness
  - Effective Hausdorff Dimension

- Applications of AIT

  - Minimum Description/Message Length
  - Machine Learning
  - Artificial Intelligence
  - Computational Complexity
  - The Incompressibility Method
  - (Shannon) information theory
  - Reversible computing
  - Universal similarity metric
  - Thermodynamics
  - Entropy and Maxwell demon
  - Compression in nature

Acknowledgements. I would like to thank Paul Vitányi for his help on improving the first draft of this article.

References

[Cal02] C. S. Calude. *Information and Randomness: An Algorithmic Perspective*. Springer, Berlin, 2nd edition, 2002.

[Cha69] G. J. Chaitin. On the length of programs for computing finite binary sequences: Statistical considerations. *Journal of the ACM*, 16(1):145–159, 1969.

[DH07] R. Downey and D. R. Hirschfeldt. *Algorithmic Randomness and Complexity*. Springer, Berlin, 2007.

[Hut05] M. Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin, 2005.

[Kol65] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information and Transmission*, 1(1):1–7, 1965.

[LV97] M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer, Berlin, 2nd edition, 1997, 3rd edition, 2007.

[Sch06] J. Schmidhuber. The new AI: General & sound & relevant for physics. In *Real AI: New Approaches to Artificial General Intelligence*. Springer, 2006.

[Sol64] R. J. Solomonoff. A formal theory of inductive inference: Parts 1 and 2. *Information and Control*, 7:1–22 and 224–254, 1964.

[ZL70] A. K. Zvonkin and L. A. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, 25(6):83–124, 1970.