MAIN Points of Algorithmic Probability

Ray Solomonoff

9/23/1985

1 What are some of its important characteristics?

1. (Important!) Definition of *randomness*. (Edit note: see Note 15 for explanation)

2. How it can be defined in terms of a universal machine or universal algorithm.

3. (Important!) P_M (Edit note: Algorithmic Probability with respect to machine M) goes from a set of data (Edit note: which can be real world data) to probability distribution.
   Kolmogorov's 1930 "Euclidification" of probability enabled probability distribution to lead to probability distribution, but *not* real world data to probability distribution.

4. Unlike the frequency concept of probability, it enables us to get reasonable probability estimates for very small samples, and even gives estimates for the probability of events that have never occurred before.

5. It has the completeness property: from sample of the output of a stochastic source, it will be able to discover the true probabilities associated with the generator. It will do this with a relatively small sample.

6. Any complete probability evaluation method must be incomputable. Conversely, any computable probability evaluation must be incomplete. It is inherently impossible to put a useful upper bound on the error of any incomplete probability evaluation method (or any *estimate* of a complete probability method.)
   In using computable approximations to P_M, the size of the error is of necessity unknown. However, we can use Algorithmic Complexity theory to give a system that gives us the *best probability* values *possible* in a small amount of computing time.
   (Edit note: The following paragraph was added from Number 14, as was more-or-less planned in there.)
   While the size of error in probability estimates can never be known with

any useful accuracy, Algorithmic Probability gives us a way to spend our time in an optimum way so that at any time our probability estimate is about the best possible.

7. An advantage of P_M over other probability evaluation methods, is that in approximating P_M, we always know what computations to do in order to attempt to improve the accuracy of our probability estimate, though we never can know how much (if at all) our probability estimate has improved.

8. P_M is a completely Bayesian method. The a priori probability on all possible objects is obtained by considering a Universal Turing Machine (or any other Universal process) that can reproduce all possible objects to be considered as its possible outputs.
   If the machine has random input, each possible output will have some probability of being generated by the machine. This probability is that which we use for the Bayesian a priori probability.
   The unavailability of this a prior probability distribution was, up to now, the only thing preventing Bayesian statistics from being a complete solution to *all* problems in statistics. Algorithmic Probability gives us this needed distribution.

9. The conditional probabilities obtained via P_M are relatively insensitive to the choice of the Universal Turing Machine used for reference. This sensitivity is further reduced if a large amount of data is used for the probability estimate.

10. One objection to the use of Algorithmic Probability is not that it is incomputable, but that it takes too long to compute i.e. its computational complexity is too high.
    However, by use of suitable training sequences and or the construction of concept nets, the computational complexity can be reduced to no more than any other probability evaluation method of equivalent "accuracy".

11. The study of concept nets and training sequences: this enables us to construct very intelligent machines - but it also enables us to understand the learning process in humans, and find ways to teach them most effectively.

12. That just about all problems can be expressed as inversion (inv) or time limited optimization (oz) problems (Edit note: See Footnote 1 for definitions of inv and oz problems).
    That Levin's Search method is an ideal way to search for an optimum solution to a problem (Edit note: See Footnote 2 for description).
    Levin's search method gives a near minimum solution time for both kinds of problems *if* all of one's knowledge about the problem is included in a suitable conditional probability distribution relating the problem to possible solutions — or alternatively all one's knowledge is in the form of a probability distribution that tells the probabilities for various choices in the maze of choices that one has in searching for a solution.

13. The concept of Conceptual Jump Size (CJS) (Edit note: See Footnote 3 for definition): Given a certain specified knowledge in a machine — both heuristic and problem-specific knowledge — CJS tells how much time it would take for that machine to solve a particular problem in a particular way - either an inv or an oz problem. It makes it possible to design training sequences and concept nets without actually running any programs.

14. This was added to Number 3, as some version of this was planned to be moved.

15. Algorithmic Probability gives us a very good understanding of randomness.
    (Edit note: added from other notes by Ray): There has been much work on development of Algorithmic Complexity to define randomness. Relatively little on using Algorithmic Complexity to define probability and then to define randomness in a simple way. An approach to defining randomness of a finite sequence is that all future continuations of the sequence are equally likely. Algorithmic Probability makes it possible to put such a definition into an exact form and analyze its properties.

Edit note: Footnotes are taken from Ray's other reports.

(Footnote 1) Inversion problems: Given a string, s and a machine, M, that maps strings to strings. Find in minimum time, a string, x, such that $M(x) = s$. Solving algebraic equations, symbolic integration and theorem proving are examples of this broad class of problems.
Time limited optimization problems: Given a time limit T, and a machine M that maps strings to real numbers, find within time T the string x such that $M(x)$ is as large as possible. Many engineering problems are of this sort — for example, designing an automobile in 6 months satisfying certain specifications having minimum cost. Constructing the best possible probability distribution or physical theory from empirical data in limited time is also of this form.

(Footnote 2) The search is done in the order of increasing amount of t/p; here t equals time needed to generate, and to test the validity, of a trial solution string of concepts, and p equals the a priori probability that the string is a correct solution. The greater the program's a priori probability, the greater is its probability of being a correct solution. (The a priori probability will be an estimate, since the true prior is incomputable). In the case of the inv problem, $M(x) = s$, a correct solution is any algorithm that can operate on both $M(\cdot)$ and s to generate x. Levin's search is closely related to the CJS of the successful solution strings (see Footnote 3). Ray's report, 'Optimal Sequential Search' (1984) (Raysolomonoff.com/pubs/opseq.pdf) gives an excellent explanation of Levin's Search, and a proof of its validity.

(Footnote 3) The CJS (conceptual jump size) of a program z is t_z divided

by p_z where p_z is the apriori probability of z and t_z is its running time. (The a priori probability will be an estimate, since the true prior is incomputable).