Algorithmic Probability — Its Discovery — Its Properties and Application to Strong AI

Ray J. Solomonoff

Visiting Professor, Computer Learning Research Centre
Royal Holloway, University of London

IDSIA, Galleria 2, CH–6928 Manno–Lugano, Switzerland
rjsolo@ieee.org http://world.std.com/~rjs/pubs.html

Introduction

We will first describe the discovery of Algorithmic Probability — its motivation, just how it was discovered, and some of its properties. Section two discusses its Completeness — its consummate ability to discover regularities in data and why its Incomputability does not hinder to its use for practical prediction. Sections three and four are on its Subjectivity and Diversity — how these features play a critical role in training a system for strong AI. Sections five and six are on the practical aspects of constructing and training such a system. The final Section, seven, discusses present progress and open problems.

1 Discovery

My earliest interest in this area arose from my fascination with science and mathematics. However, in first studying geometry, my interest was more in how proofs were discovered than in the theorems themselves. Again, in science, my interest was more in how things were discovered than in the contents of the discoveries. The Golden Egg was not as exiting as the goose that laid it.

These ideas were formalized into two goals: one goal was to find a general method to solve all mathematical problems. The other goal was to find a general method to discover all scientific truths. I felt the first problem to be easier because mathematics was deterministic and scientific truths were probabilistic. Later, it became clear that the solutions to both problems were identical![1]

[1]The subject of the beginning of this essay has been treated in some detail in "The Discovery of Algorithmic Probability"(Sol 97). Here, we will summarize some ideas in that paper and deal with important subsequent developments.

Some important heuristic ideas:

First — From Rudolph Carnap: That the state of the universe could be represented by a long binary string, and that the major problem of science was the prediction of future bits of that string, based on its past.

Second — From Marvin Minsky and John McCarthy: the idea of a universal Turing machine. That any such machine could simulate any describable function or any other Turing machine (universal or not). That it had a necessary "Halting Problem" — that there had to be inputs to the machine such that one could never be sure what the output would be.

Third — Noam Chomsky's description of Generative and Nongenerative grammars. To use them for prediction, in 1958 I invented Probabilistic Grammars (Described in the appendix of (Sol 59)).

The final discovery occurred in 1960 (Sol 60) when I began investigating the most general deterministic Grammar — based on a universal Turing machine. It's probabilistic version had some striking properties and suggested to me that a probabilistic grammar based on a universal Turing machine would be the most general type of grammar possible — and would perhaps be the best possible way to do prediction.

This was the birth of Algorithmic Probability (ALP). In the initial version, we take a universal Turing machine with an input tape and an output tape. Whenever the machine asks for an input bit, we give it a zero or a one with probability one–half. The probability that (if and when the machine stops) the output tape is a certain binary string, x, is the universal probability of x. This was a universal distribution on finite strings.

I was much more interested in sequential prediction (as in Carnap's problem), so I generalized it in the following way: We use a special kind of universal Turing machine. It has three tapes — unidirectional input and output tapes, and an infinite bidirectional work tape. We populate the input tape with zeros and ones, each with probability one–half. The probability of the string x is the probability that the output tape will be a string that begins with x.

This second universal distribution can be used for sequential prediction in the following way: suppose $P(x1)$ is the probability assigned by the distribution to the string, $x1$. Let $P(x0)$ be probability assigned to $x0$. Then the probability that x will be followed by 1 is

$$P(x1)/((P(x0) + P(x1)) \tag{1}$$

I will be usually be referring to this second model when I discuss ALP.

It is notable that ALP doesn't need Turing machines to work properly. Almost all of its properties carry over if we use any computer or computer language that is "universal" — i.e. that it can express all computable functions in an efficient way. Just about all general purpose computers are "universal" in this sense, as are general programming languages such as Fortran, LISP, C, C++, Basic, APL, Mathematica, Maple, ...

2 Completeness and Incomputability

Does ALP have any advantages over other probability evaluation methods? For one, it's the only method known to be *complete*. The completeness property of ALP means that if there is any regularity in a body of data, our system is guaranteed to discover it using a relatively small sample of that data. More exactly, say we had some data that was generated by an *unknown* probabilistic source, P. Not knowing P, we use instead, P_M, the Algorithmic Probabilities of the symbols in the data. How much do the symbol probabilities computed by P_M differ from their true probabilities, P?

The Expected value with respect to P of the total square error between P and P_M is bounded by $-1/2\, lnP_0$.

$$E_P\Big[\sum_{m=1}^{n}(P_M(a_{m+1}=1|a_1,a_2\cdots a_m)-P(a_{m+1}=1|a_1,a_2\cdots a_m))^2\quad\Big]\leq\quad -\frac{1}{2}\,lnP_0$$

$$lnP_0\approx kln2 \tag{2}$$

P_0 is the a priori probability of P. It is the probability we would assign to P if we knew P.

k is the *Kolmogorov complexity* of the data generator, P. It's the shortest binary program that can describe P, the generator of the data.

This is an extremely small error rate. The error in probability approaches zero more rapidly than $1/n$. Rapid convergence to correct probabilities is a most important feature of ALP. The convergence holds for any P that is describable by a computer program and includes many functions that are formally *incomputable*. The convergence proof is in (Sol 78). It was discovered in 1968, but since there was little general interest in ALP at that time I didn't publish until 1975 (Sol 75) and it wasn't until 1978 (Sol 78) that a proof was published. The original proof was for a mean square loss function and a normalized universal distribution. — but the proof itself showed it to be also true for the more general KL loss function. Later, Peter Gács (Gács 97) showed it would work for a universal distribution that was not normalized and Marcus Hutter (Hut 02) showed it would work for arbitrary (non-binary) alphabets, and for a variety of loss functions.

While ALP would seem to be the best way to predict, the scientific and mathematical communities were disturbed by another property of algorithmic probability: — it was incomputable! This incomputability is attributable to "the halting problem" — that there will be certain inputs to the Turing machine for which we can never be certain as to what the output will be.

It is, however, possible to get a sequence of approximations to ALP that converge to the final value, but at no point can we make a useful estimate as to how close we are to that value.

Fortunately, for practical prediction, we very rarely need to know "the final value". What we really need to know is "How good will the present approximate predictions be in the future (out of sample) data"? This problem occurs in all

prediction methods and algorithmic probability is often able to give us insight on how to solve it.

It is notable that completeness and incomputability are complementary properties: It is easy to prove that any *complete* prediction method must be incomputable. Moreover, any computable prediction method cannot be *complete*— there will always be a large space of regularities for which its predictions are catastrophically poor.

Since incomputability is no barrier to practical prediction, and computable prediction methods necessarily have large areas of ineptitude, it would seem that ALP would be preferred over any computable prediction methods.

There is, however another aspect of algorithmic probability that people find disturbing — it would seem to take too much time and memory to find a good prediction. In Section 5 we will discuss this at greater length. There is a technique for implementing ALP that seems to take as little time as possible to find regularities in data.

3 Subjectivity

Subjectivity in science has usually been regarded as Evil. — that it is something that does not occur in "true science" — that if it does occur, the results are not "science" at all. The great statistician, R. A. Fisher, was of this opinion. He wanted to make statistics "a true science" free of the subjectivity that had been so much a part of its history.

I feel that Fisher was seriously wrong in this matter, and that his work in this area has profoundly damaged the understanding of statistics in the scientific community — damage from which it is recovering all too slowly.

Two important sources of error in statistics are finite sample size and model selection error. The finite sample part has been recognized for some time. That model selection error is a necessary part of statistical estimation is an idea that is relatively new, but our understanding of it has been made quite clear by ALP. Furthermore, this kind of error is very subjective, and can depend strongly on the lifelong experience of a scientist.

In ALP, this subjectivity occurs in the choice of "reference" — a universal computer or universal computer language. In the very beginning, (from the "invariance theorem") it was known that this choice could only influence probability estimates by a finite factor — since any universal device can simulate any other universal device with a finite program. However, this "finite factor" can be enormous — switching between very similar computer languages will often give a change of much more than 2^{1000} in probability estimates!

To understand the role of subjectivity in the life of a human or an intelligent machine, let us consider the human infant. It is born with certain capabilities that assume certain a priori characteristics of its environment–to–be. It expects to breathe air, its immune system is designed for certain kinds of challenges, it is usually able to learn to walk and converse in whatever human language it finds in its early environment. As it matures, its a priori information is modified

and augmented by its experience.

The AI system we are working on is of this sort. Each time it solves a problem or is unsuccessful in solving a problem, it updates the part of its a priori information that is relevant to problem solving techniques. In a manner very similar to that of a maturing human being, its a priori information grows as the life experience of the system grows.

From the foregoing, it is clear that the subjectivity of algorithmic probability is a necessary feature that enables an intelligent system to incorporate experience of the past into techniques for solving problems of the future.

4 Diversity

In Section 1 we described ALP with respect to a universal Turing machine with random input. An equivalent model considers all prediction methods, and makes a prediction based on the weighted sum of all of these predictors. The weight of each predictor is the product of two factors: the first is the a priori probability of each predictor. — It is the probability that this predictor would be described by a universal Turing machine with random input. If the predictor is described by a small number of bits, it will be given high a priori probability. The second factor is the probability assigned by the predictor to the data of the past that is being used for prediction. We may regard each prediction method as a kind of model or explanation of the data. Many people would use only the best model or explanation and throw away the rest. Minimum Description Length (Ris 78), and Minimum Message Length(Wal 68) are two commonly used approximations to ALP that use only the best model of this sort. When one model is much better than any of the others, then Minimum Description Length and Minimum Message Length and ALP give about the same predictions. If many of the best models have about the same weight, then ALP gives better results.

However, that's not the main advantage of ALP's use of a diversity of explanations. If we are making a single kind of prediction, then discarding the non-optimum models usually has a small penalty associated with it. However if we are working on a sequence of prediction problems, we will often find that the model that worked best in the past, is inadequate for the new problems. When this occurs in science we have to revise our old theories. A good scientist will remember many theories that worked in the past but were discarded — either because they didn't agree with the new data, or because they were a priori "unlikely". New theories are characteristically devised by using failed models of the past, taking them apart, and using the parts to create new candidate theories. By having a large diverse set of (non-optimum) models on hand to create new trial models, ALP is in the best possible position to create new, effective models for prediction.

When ALP is used in Genetic Programming, it's rich diversity of models can be expected to lead to very good, very fast solutions with little likelihood of "premature convergence".

5 Computation Costs

If Algorithmic Probability is indeed so very effective, it is natural to ask about its computation costs — it would seem that evaluating a very large number of prediction models would take an enormous amount of time. We have, however, found that by using a search technique similar to one used by Levin for somewhat different kinds of problems, that it is possible to perform the search for good models in something approaching optimum speed. It may occasionally take a long time to find a very good solution — but no other search technique could have found that solution any faster.

A first approximation of how the procedure works: Suppose we have a universal machine with input and output tapes and a very big internal memory. We have a binary string, x, that we want to extrapolate — to find the probability of various possible continuations of x. We could simply feed many random strings into the machine and watch for inputs that gave outputs that started with x. This would take a lot of time, however. There is a much more efficient way:

We select a small time limit, T, and we test all input strings such that

$$t_k < T2^{-l_k} \tag{3}$$

Here l_k is the length of the k^{th} input string being tested, and t_k is the time required to test it. The test itself is to see if the output starts with string x. If we find no input that gives output of the desired kind, we double T and go through the same test procedure. We repeat this routine until we find input strings with output strings that start with x. If we give each such output a weight of 2^{-l_k} (l_k being the length of its input), the weighted output strings will get a probability distribution over the possible continuations of the string, x.

In the example given, all input strings of a given length were assumed to have the same probability. As the system continues to predict a long binary sequence, certain regularities will occur in input sequences that generate the output. These regularities are used to impose a nonuniform probability distribution on input strings of a given length. In the future this "adaptation" of the input distribution enables us to find much more rapidly, continuations of the data string that we want to predict.

6 Training Sequences

It is clear that the sequence of problems presented to this system will be an important factor in determining whether the mature system will be very much more capable than the infant system. Designing "training sequences" of this sort is a crucial and challenging problem in the development of strong intelligence.

In most ways, designing a training sequence for an intelligent machine, is very similar to designing one for a human student. In the early part of the sequence, however, there is a marked difference between the two. In the early training sequence for a machine, we know exactly how the machine will react

to any input problem. We can calculate a precise upper bound on how long it will take the machine to solve early problems. It is just

$$T_i/P_i \tag{4}$$

P_i is the probability that the machine assigns to the solution that is known by the trainer. T_i is the time needed to test that solution. I call this upper bound the "conceptual jump size" (CJS). It tells us how hard a problem is for a particular AI. I say "upper bound" because the system may discover a better, faster, solution than that known by the trainer.

This CJS estimate makes it easy to determine if a problem is feasible for a system at a particular point in its education. The P_i for a particular problem will vary during the life of the system. For a properly constructed training sequence, the P_i associated with a particular problem should increase as the system matures.

Eventually in any training sequence for a very intelligent machine, the trainer will not be able to understand the system in enough detail to compute CJS values. The trainer will then treat the machine as a human student. By noting which problems are easy and which are difficult for the machine the trainer will make a very approximate model of the machine and design training problems using that model.

Learning to train very intelligent machines should give very useful insights on how to train human students as well.

7 Where Are We Now?

A system incorporating some of the features we have discussed has been programmed by Schmidhuber (Sch 02). It was able to discover a recursive solution to the "Tower of Hanoi" problem, after finding a recursive solution to one of its earlier, easier problems.

For further progress, we need larger, more detailed training sequences — Writing sequences of this sort is a continuing "Open Problem" (Sol 89)

The process of updating the system in view of its past experience is another important area of ongoing research. We have considered PPM (Prediction by Partial Matching (Tea 95)), APPM (an augmented version of PPM) and SVR (Support Vector Regression (Sap 09)) as possible updating systems. The improvement of the updating algorithm remains another continuing "Open Problem".

References

[1] (Gács 74) Gács. P. Theorem 5.2.1 in *An Introduction to Kolmogorov Complexity and Its Applications*, Springer–Verlag, N.Y., 2nd edition, pp. 328-331, 1997.

[2] (Hut 02) Hutter, M.,"Optimality of Universal Bayesian Sequence Prediction for General Loss and Alphabet," http://www.idsia.ch/ marcus/ai/

[3] (Ris 78) Rissanen, J. "Modeling by the Shortest Data Description," *Automatica*, 14:465–471, 1978.

[4] (Sch 02) Schmidhuber, J., "Optimal Ordered Problem Solver," TR IDSIA-12-02, 31 July 2002. http://www.idsia.ch/ juergen/oops.html

[5] (Sap 09) Sapankevych, N. and Sankar, R., "Time Series Prediction Using Support Vector Machines: A Survey," *IEEE Computational Intelligence* Vol. 4, No. 2, pp 24–38, May 2009

[6] (Sol 59) Solomonoff, R.J. "A Progress Report on Machines to Learn to Translate Languages and Retrieve Information," *Advances in Documentation and Library Science*, Vol. III, pt. 2, pp. 941–953. (Proceedings of a conference in September 1959.)

[7] (Sol 60) Solomonoff, R.J. "A Preliminary Report on a General Theory of Inductive Inference." (Revision of Report V–131, Feb. 1960), Contract AF 49(639)–376, Report ZTB–138, Zator Co., Cambridge, Mass., Nov, 1960.

[8] (Sol 75) Solomonoff, R.J. "Inductive Inference Theory – a Unified Approach to Problems in Pattern Recognition and Artificial Intelligence," *Proceedings of the 4th International Conference on Artificial Intelligence*, pp 274––280, Tbilisi, Georgia, USSR, September 1975.

[9] (Sol 78) Solomonoff, R.J. "Complexity–Based Induction Systems: Comparisons and Convergence Theorems," *IEEE Trans. on Information Theory*, Vol IT—24, No. 4, pp. 422 –432, July 1978.

[10] (Sol 89) Solomonoff, R.J. "A System for Incremental Learning Based on Algorithmic Probability," *Proceedings of the Sixth Israeli Conference on Artificial Intelligence, Computer Vision and Pattern Recognition,* pp. 515–527, Dec. 1989.

[11] (Sol 97) Solomonoff, R.J. "The Discovery of Algorithmic Probability," *Journal of Computer and System Sciences*, Vol. 55, No. 1, pp. 73–88, August 1997.

[12] (Tea 95) Teahan, W.J. "Probability Estimation for PPM," *Proc. of the New Zealand Computer Science Research Students' Conference*, University of Waikato, Hamilton, New Zealand, 1995. http://cotty.16x16.com/compress/peppm.htm

[13] (Wal 68) Wallace, C.S and Boulton, D.M. "An Information Measure for Classification," *Computer Journal*, 11:185–194, 1968.