A Few Notes on Multiple Theories and Conceptual Jump Size

Grace Solomonoff

Oxbridge Research
Mailing Address: P.O.B. 400404, Cambridge, Ma. 02140, U.S.A.
prettyvivo@gmail.com http://raysolomonoff.com

Abstract. These are a few notes about some of Ray Solomonoff's foundational work in algorithmic probability, focussing on the universal prior and conceptual jump size, including a few illustrations of how he thought. His induction theory uses Bayes rule to measure and compare the likelihood of different theories describing observations. Can we find good theories? Lsearch is a very good way to search and the conceptual jump size is a measure for this.

1 Understanding and Learning

1. Ray taking apart a scooter. 2. Working on the house he built. 3. An "experiment" (Ray is on the right).

The first thing Ray did when he acquired something was to take it apart. Here's a picture of Ray taking apart a scooter he found in the trash. He took many notes. They were like a program, so that the scooter could be remade. Ray built a house from descriptions in a book, like a recipe. What was made in Ray's lab in the cellar?

Ray [Sol97] wrote:

> My earliest contact with modern scientific philosophy may have been P.W. Bridgman's [Bri27] concept of "operational definition". An operational definition of anything is a precise sequence of physical operations that enable one to either construct it or identify it with certainty.

> ... When one can't make an operational definition of something, this is usually an indication of poor understanding of it. ... Attempts to operationalize definitions can be guides to discovery. I've found this idea to be an invaluable tool in telling me whether or not I really understand something.

To quantify information Solomonoff's theory uses operational definitions by means of computer programs.

A new way to measure things may herald a new era in math or science. It can bring new ways to see, and new tools. A new way to measure information content of a string happened in 1960-65 when Ray Solomonoff (60, 64) [Sol60b,Sol64], Andrey Kolmogorov (65) [Kol65] and Gregory Chaitin (66) [Cha66] independently published a new way to measure the complexity of a sequence of observations by the size of the minimal computer program that could produce it. Kolmogorov and Chaitin were interested in the descriptional complexity of a sequence: to quantify the information, and use that to define randomness, while Ray was interested in the prediction aspects of the sequence, and to compare, using an accurate measure, how well different theories (expressed as computer programs) could predict future members of a sequence.

Prediction and learning are closely related. The heart of science is prediction and the likelihood of that prediction.[1]

I think Ray thought the length and number of explanations that produced or described a sequence of observations was related to learning. He expressed these explanations as computer programs. We don't yet know if this new way to measure will be important, but there is a good possibility.

It implies that understanding and learning are not unknowable things trapped in the brain's black box; they may be weird, but they will be understood.

Ray used a method called algorithmic probability (AP) to measure, and add programs (theories, algorithms). He used a method of searching for theories called Lsearch which is related to a measure he called conceptual jump size (CJS). Lsearch, also called Levin search is named for Leonid Levin who gave it mathematical precision.

This paper will discuss my view of a few of Solomonoff's ideas about AP, Lsearch and CJS. This interpretation is simply from an interest and enjoyment I had from hearing him talk about his work. He had delight in creating something new, a joy that is there for all who search for new ideas. Hopefully his ideas will contribute to his lifelong interest of achieving a thinking machine able to solve hard problems in all domains.

[1] Earlier Ray wondered [Sol03] why Kolmogorov wasn't interested in using these concepts for inductive inference — to define empirical probability. "Leonid Levin suggested that inductive inference was at that time, not actually a "mathematical" problem... It may have been that in 1965 there was no really good definition of induction and certainly no general criterion for how good an inductive system was." In that paper Ray also points out the difference in meaning of "Universal" between the universal distribution on one Turing Machine and dependence of the universal distribution on choice of machine.

2 Algorithmic Probability and The Suite of Theories

In a letter in 2011, Marcus Hutter wrote: "Ray Solomonoff's universal probability distribution M(x) is defined as the probability that the output of a universal monotone Turing machine U starts with string x when provided with fair coin flips on the input tape. Despite this simple definition, it has truly remarkable properties, and constitutes a universal solution of the induction problem."(See also [RH11])

To predict the continuation of a string, x, AP gives weights to the theories, expressed as computer programs, that could produce or describe it:

$$P_M(x) = \sum_{i=1}^{\infty} 2^{-|s_i(x)|} \tag{1}$$

It measures the likelihood of each unique, randomly created program s_i input into Machine M to produce a chosen sequence, x; this means all the different ways that x could be replicated. Then it adds these probabilities together. The resulting value is P probability of the sequence with respect to Machine M.

The shortest program in this distribution is intuitively like the idea of Occam's Razor: the simplest program is the best. Using all the individual programs is intuitively like an idea of Epicurus: keep all of the theories. But the simpler theories are more likely. It weights each theory by a measure based on the likelihood of that theory.

The Universal Prior by its sum defines the probability of a sequence, and by using the weight of individual programs gives a figure of merit to each program (cause or description – like an operational definition) that could produce the sequence [Sol64]. He uses Bayes' rule to predict the different continuations of the sequence and the likelihood of each continuation. Thus probability ratios can be established using this system.

The simplest explanation for an event is often the best one. But something is also more likely to happen if there are many possible causes. Some researchers use AP to choose the single shortest program for prediction [WD99]. Ray thought that keeping as many programs as possible is best for some situations [Sol84]:

> Why not use the single "best" theory? The best is via one criterion; a) i.e. min a priori of theory x (pr [probability] of corpus with right theory); b) however another best is with respect to b) minimum expected prediction error. For best in b) we use weights of various theories, using weights of a)

Any new hypothesis can be measured and added to the group of hypotheses; in a finite world, we don't have to include every possible hypothesis from an infinite amount.

3 Using Bayes' rule

Bayes' rule is used in many statistical applications to sort out frequencies of events. If the events are called causes then Bayes' rule becomes "Bayes' rule for the probability of causes" [Fel50].

Bayes' rule may be the optimal method of predicting future observations based on current beliefs and evidence. A value of AP is that it provides the necessary prior, the universal prior, that contains the complete universe of theories used for prediction. When little or no prior information is available, this technique enables us to construct a default prior based on the language used to describe the data.

The shortest input computer programs describing the sequence x are more likely than the longer ones, but there will always be at least one theory that can describe any finite sequence of observations: this input simply duplicates every observation one by one, which as a program, translates to "print $< x >$" for any given sequence x. Thus none of the sequences will have zero probability. It avoids the problem of having to select something randomly[Fel50], for the theories will have a default order based on program length complexity.

Ray [Sol99] wrote: "If there is any describable regularity in a corpus of data, this technique will find the regularity using a relatively small amount of data. – While the exact a priori probability is mathematically incomputable, it is easy to derive approximations to it. An important outgrowth of this research has been the insight it has given into the necessary trade-offs between a priori information, sample size, precision of probability estimates and computation cost.".

In early years probability was scorned in the AI community. Nowdays, in Artificial Intelligence (AI), Bayes' rule is usually used for frequencies, – sorting out masses of data; it gives good statistical values. The frequency version deals with a search space like a Greek urn with events that are variously colored balls. Bayes' rule helps sort the events into groups that may be within other groups, relating them to a common base, so you can add ratios together properly.

I think Ray changed the Greek urn into a Turing urn filled with events that are hypotheses. In this urn are explanations, not objects. The explanations may be different, but may begin the same way.

In a letter by Alex Solomonoff [Sol16b], Alex remembers Ray telling him that perhaps all probability is causal, not frequentist – a coin has a 50% chance of coming up heads only because we are ignorant of the details of how it is flipped. Also Alex notes that in Solomonoff's theory, the events or observations being predicted by AP are deterministic, not random. The Universal prior implies that the universe has structure and can be described by rules, *not* derived from frequencies of events.

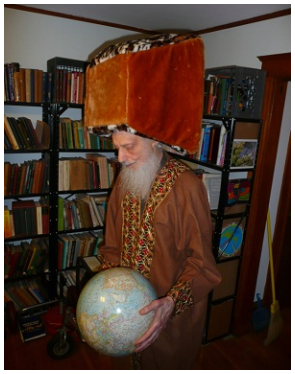
4 Incomputability

AP is (almost) as "computable" as the value of π with successive approximations that are guaranteed to converge to the right value. But unlike π, we can not know how large the error in each approximation can be.

Ray [Sol97] wrote, "The question of the "validity" of any inductive inference methods is a difficult one. You cannot prove that any proposed inductive inference method is "correct.", only that one is "incorrect" by proving it to be internally inconsistent, or showing that it gives results that are grossly at odds with our intuitive evaluation."

But Ray often said about incomputibility: "It's not a bug, it's a feature!" Systems that are computable cannot be complete [Leg06]. Incomputability is because some algorithms can never be evaluated because it would take too long. But these programs will be recognized as possible solutions. On the other hand, any computable system is incomplete: there will always be regularities outside system's search space which will never be acknowledged or considered, even in an infinite amount of time. Computable prediction models hide this fact by ignoring such descriptions.

5 Metamorphoses of a Theory


1. A Sultan in a turban. 2. Ray in a Turban 3. Mimi the cat in her house.

The second thing Ray did when he acquired something was to use it in a way for which it wasn't intended. When Ray discovered a new use for Mimi the cat's house, he described the use of Mimi's house in a new way; a container with an opening, which can be used in different way. The new encompassing theory leads to greater flexibility, more uses for Mimi's (who wasn't all that happy about this!) house.

A different view may help making or attempting predictions. One day Ray turned the graph of the Dow Jones index upside down. It looked different, which indicated that there was information there. Another day he got my brother to try "playing" several indices on his viola.

In general, a new description is evidence of a kind of learning. Ray uses a method called Lsearch to look for descriptions and a measure called conceptual Jump Size to quantify steps of this learning.

6 Lsearch

Lsearch is a computable way to build simple theories that match the observations. Ray [Sol97] explained it in a biographical article:

"In the present context, any "concept" can be represented by string of computer instructions – a "macro". They are combined by concatenation.

Given a machine, M, that maps finite strings onto finite strings. Given the finite string, x. How can we find in minimal time, a string, p, such that $M(p) = x$?

Suppose there is an algorithm, A, that can examine M and x, then print out p within time T. Levin had a search procedure that, without knowing A, could do the same thing that A did, but in no more time than $CT2^L$. Here, L is the length of the shortest description of A, using a suitable reference machine, and C is a measure of how much slower the reference machine is than a machine that implements A directly. An alternative form of this cost of the search is CT/P. Here $P = 2^{-L}$ is approximately the a priori probability of the algorithm, A.

The parameter T/P plays a critical role in searches of all kinds. In designing sequences of problems to train an induction machine the T/P value of a particular problem at a particular point in the training of the machine gives an upper bound on how long it will take the machine to solve the problem. In analogy with human problem solving, I christened T/P "Conceptual Jump Size".

Before I met Levin, I had been using T/P as a rough estimate of the cost of a search, but I had no proof that it was achievable. . . . Sometime later, I showed that Levin's search procedure (which I will henceforth denote by "Lsearch") is very close to optimum if the search is a "Blind Search""

Lsearch may take the first program to match observations, so it is close to the spirit of Kolmogorov complexity, though there are versions that will search for more than one program. Lsearch hasn't been applied very much for real problem solving. The measure gives an upper bound to how much time it will take, but it does not tell about error size, and the bound though finite can be very large. Perhaps Schmidhuber [Sch94] was the first to run a computer program using a very simple Lsearch to solve a problem. Ray thought Lsearch could be used.

7 Conceptual Jump Size and Descriptions

CJS is related to the difference of Kolmogorov complexity of a growing string of observations, where the computation time to find a new best description is taken into account. In schools, most often, problems are given that are either right or wrong. But in the real world, plans that first seemed right often fail when new data comes in: so we have Plan B and Plan C, which in Ray's work, may be represented by the members of the suite of programs. Conceptual jump size may give a way to think about questions like this. For example if the sequence is "*water water water. . .*", then the shortest code is likely "repeat *water* forever". However, if the sequence is "*water water water. . . chicken chicken. . . [some differential equation]. . . [a game of chess]. . .*", then the simplest description may become more and more complicated as the sequence progresses. Each time there

is a lack of regularity between each observation there is a jump of complexity of the description, resulting in a jump of (decreasing) probability of the sequence. The conceptual jump size is a computable way to measure how much more complex the sequence becomes.

8 Can the Search be practical?

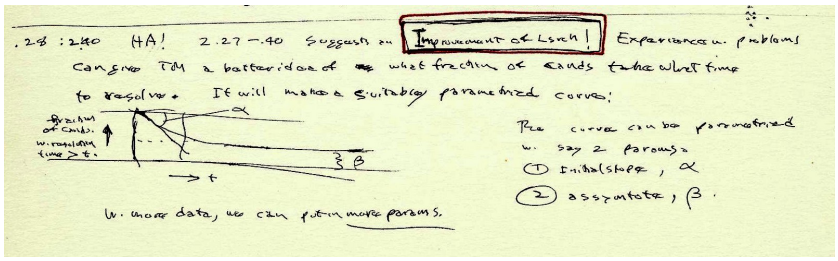
"Nothing is always absolutely so" (T. Sturgeon, July 57): was Ray's comment on the cover of his first 1960 Inductive Inference report [Sol60a].

A big problem is that a search for better theories may take more time than can ever be available. Yet people find new theories all the time. If people can do it, so can AI. Can CJS be brought into reasonable size?

Ray had other ideas about Lsearch; for example: improving it by altering the next cycle of Ti, so that the same codes wouldn't be found a second time. Much time is wasted if Lsearch doesn't remember what it did in earlier cycles.

In another note, a question... "The question about if I've spent time T_0 seconds on compression without more in compression what is probability of finding more compression in next t seconds. Is it t/t_0 or $1-(t/(epsilon_0^t))$.[Sol84]"

Among the notes, a small graph hints toward another idea, perhaps the slopes are part of comparing oversearching:



Ray was interested in Lcost as a part of Lsearch which incorporated the cost of computer activity; so there would be penalty for nonproductive activity caused by a successful code. A similar measure of learning is action. The use of dollar cost (in those days equipment cost x rental time) is a good cost measure for most AI type calculations. "Action is approximately equal to equipment cost X time".[Sol81]

Ray also developed the idea of RLP (Resource Limited Prediction) to deal with four basic factors: They are: 1.) The prediction itself. 2.) The reliability of that prediction. 3.) The sample size. 4.) Computation cost.[Sol81]

See raysolomonoff.com/raysnotes/ for Ray's handwritten review of Lsearch (1979).

Here are some more ideas by Alex Solomonoff [Sol16a]:

Suppose there are two codes, (with different continuations) that both reproduce the observed data. Both are the same length, and both are the shortest

code in their respective cloud of functionally equivalent codes.[2] Then the code with the bigger, denser cloud will be more probable.

Training sequences were a method Ray Solomonoff suggested for leading the Lsearch to the destination in steps short enough to be practical. The idea of a training sequence suggests Levin search can be taught things the trainer already knows. But how would it learn things that no-one knows? You would run short Levin searches on every bit of data you could find. Occasionally you might find a regularity. Those few short codes, and that data would be your training sequence. You wouldn't require that the short code reproduce the entire string perfectly. If it got "enough" of it right, you would call it a success. But this requires that a large corpus of data be divided up into bite-sized chunks, and there are a million ways to do this. Even if there were a "natural" way of splitting it, how would the machine find it? More undirected Lsearch. This would be a very slow process.

In the most basic Levin search, the CPU fraction assigned to a code is determined solely by the length of the code.

1. If a running code has not finished printing out x, and has not printed out a bit in a long time, it is probably in an infinite loop, will never print out another bit, so we should reduce its CPU fraction.

2. If a running code has correctly printed out most of the bits of x, it is more likely to output all of the bits correctly than a code that has only printed out a few bits. So we should increase the CPU fraction of a code every time it prints out a correct bit.

9 Agents

Agents are also being developed to speed up the search.

Marcus Hutter's general concept of an agent is one that, in the scientific world, can make experiments to get more meaningful. observations faster than waiting for the universe to provide them. Laurent Orseau and Tor Lattimore and Marcus Hutter [OLH13] developed a Bayesian knowledge-seeking agent.

These are a few examples of ways to shorten CJS size.

10 Fun with Unconscious Jumps

Ray asked how people do learning jumps – mostly by unconscious methods.

Pac-Man interested him, because a person plays and plays and does maybe a bit better, and then suddenly does much better. Ray believed that was the unconscious making a good jump to a new method of play. Nobody yet knows how the jump occurs but the action shows that it did occur. It remains in the

[2] Two codes are functionally equivalent if all the bits they ever output are identical. If two codes compute all the same output bits, how they generate those bits won't make any difference to any prediction or probability. Except for the matter of computing speed, in a time limited code search...

players unconscious as a tool, and using it, may lead on to the next level. Unless a method is remembered it isn't learned.

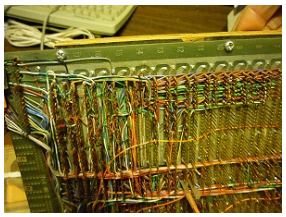
Jokes are a mental kind of sports; sports are a fun way to make people better at hunting and other physical survival skills. Finding punchlines to jokes make people better at discovering interesting theories — a mental survival skill — they are a fun way to enlarge our ability to jump from one theory to another: there is a normal description of something, and then in the joke we get a nifty different description: a 'funny' description that is just right, and the reward is that we 'get it!' and laugh. So in this way we may learn more from jokes that encourage our ability to leap to better theories, than from the school homework that has Yes or No answers.

Many theories; simple theories: what kind of probability has the Universe?

11 On the back porch just beyond the Universe

Kolmogorov and Solomonoff were sitting admiring the view when Kolmogorov brought out a string of 2^{1000} bits and set it on the wicker table next to his Kvas. He took out his Universal Turing Rover laptop and in 10 seconds found one program of 100 bits that described his string.

Solomonoff set down his root beer, and brought out a string of 2^{1000} bits and set it on the table. He took out his Universal Turing Handmade computer[3] and in 10 seconds found 2^{50} programs that described it, each program of 110 bits.


1.,2. Ray's handmade computer 3. Looking into space during fireworks

Andrey said, "My string seems least random because my program is only 100 bits, while yours are 110 bits." Ray said, "Mine seems so because your single program of 100 bits has prior probability of 2^{-100}, while my 2^{50} programs give a combined probability of 2^{50} x 2^{-110} which is 2^{-60}"

Solomonoff turned to Kolmogorov, and added, "I use shortest codes to measure my hypotheses and in Lsearch."

[3] Of course they both had the same Richard Stallman [free] v. 4.3 computer architecture instruction set

In our finite world in short periods of time, perhaps multiple codes may have more weight than a single code. But what about a longer and longer string as seen from the back porch?

Ray [Sol64] said, Hmm, "...if T is a very long string, ... $P_i(T)$ normally decreases exponentially as T increases in length. Also, if Q_i and Q_j are two different probability evaluation measures (PEM's) and Q_i is "better" than Q_j, then usually $P_i(T)/P_j(T)$ increases exponentially as T increases in length. Of greater import, however, ... the relative weight of Q_i and Q_j, increases to arbitrarily large values for long enough T's. This suggests that for very long T, ... [AP] ... gives almost all of the weight to the single "best" PEM.".

Sitting on the back porch, just beyond the Universe, Kolmogorov turned to Solomonoff.

He said "Almost?!?"

References

[Bri27] P.W. Bridgman. *The Logic of Modern Physics.* Macmillan Company, New York, 1927.

[Cha66] G.W. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the Assoc. of Comp. Mach.*, (13):547–569, 1966.

[Fel50] W. Feller. *Probability Theory and Its Applications.* Wiley and Sons, 1950.

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

[Leg06] S. Legg. Is there an elegant universal theory of prediction. In J. Balczar, P. Long, and F. Stephan, editors, *Proc. 17th International Conf. on Algorithmic Learning Theory (ALT'06)*, volume 8139 of *LNAI*, pages 274–287, Barcelona, October 2006. Springer.

[OLH13] Laurent Orseau, Tor Lattimore, and Marcus Hutter. Universal knowledge-seeking agents for stochastic environments. In S. Jain, R. Munos, F. Stephan, and Th. Zeugmann, editors, *Proc. 24th International Conf. on Algorithmic Learning Theory (ALT'13)*, volume 8139 of *LNAI*, pages 158–172, Singapore, October 2013. Springer.

[RH11] S. Rathmanner and M. Hutter. A philosophical treatise of universal induction. *Entropy*, (13):1076–1136, 2011.

[Sch94] J. Schmidhuber. Discovering problem solutions with low kolmogorov complexity and high generalization capability. Technical report, Technische Universität, München, 1994.

[Sol60a] R.J. Solomonoff. A preliminary report on a general theory of inductive inference. Technical Report V-131, Zator Co. and Air Force Office of Scientific Research, Cambridge, Mass., Feb 1960.

[Sol60b] R.J. Solomonoff. A preliminary report on a general theory of inductive inference. (revision of Report V-131). Technical Report ZTB-138, Zator Co. and AFOSR, Cambridge, Mass., Nov 1960.

[Sol64] R.J. Solomonoff. A formal theory of inductive inference: Part I. *Information and Control*, 7(1):1–22, March 1964.

[Sol81] R.J. Solomonoff. Notes for mit talk. Solomonoff Archive, November 1981.

[Sol84] R.J. Solomonoff. Notes for harvard talk, 1984.

[Sol97] R.J. Solomonoff. The discovery of algorithmic probability. *Journal of Computer and System Sciences*, 55(1):73–88, August 1997.
[Sol99] R. J. Solomonoff. A letter to david malkoff, Feb 1999.
[Sol03] R.J. Solomonoff. The universal distribution and machine learning. *The Computer Journal*, (6):508–601, 2003.
[Sol16a] A Solomonoff. Algorithmic probability. Privately circulated report, 2016.
[Sol16b] A. Solomonoff. A letter. Personal Communication, February 2016.
[WD99] C.S. Wallace and D. L. Dowe. Minimum message length and Kolmogorov complexity. *Computer Journal*, 42(4):270–283, 1999.