TWO KINDS OF PROBABILISTIC INDUCTION

Ray Solomonoff

Visiting Professor, Computer Learning Research Center
Royal Holloway, University of London
Mailing Address: P.O.B. 400404, Cambridge, Ma. 02140, U.S.A.
rjsolo@ieee.org

Abstract

Problems in probabilistic induction are of two general kinds. In the first, we have a linearly ordered sequence of symbols that must be extrapolated. In the second we want to extrapolate an unordered set of finite strings.

A very general formal solution to the first kind of problem is well known and much work has been done in obtaining good approximations to it [LI 93, Ris 78, Ris 89, Sol 64a, Sol 64b, Sol 78, Wal 68].

Though the second kind of problem is of much practical importance, no general solution has been published. We present two general solutions for unordered data.

We also show how machines can be constructed to summarize sequential and unordered data in optimum ways.

1   Introduction

In the early days of Algorithmic Probability , we felt that all problems in prediction could be put in the form of the prediction of a sequence of digital symbols [Sol 64a, p.2]. Though Algorithmic Probability solved the sequential prediction problem, it hasn't been clear as to how it could be applied to many problems in induction that did *not appear* to be sequential. The present paper shows how to do this.

There are at least two essentially different kinds of problems that occur in prediction. We will describe them and contrast the ways in which they can be treated.

The first kind of problem is sequential prediction. We have a sequence of symbols starting at some point and extending (possibly) infinitely. Given a finite prefix, D, of this sequence, to predict the most likely continuation and/or the probabilities of various possible continuations. An example might be predicting tomorrow's stock price, given the previous historical daily prices of that stock.

The data involved can be all numerical, but it may include discrete information such as newspaper articals or other news of the changing state of the world. Sequential data can always be represented by a finite string of symbols.

The second kind of problem is prediction of unordered data. We have an unordered set of n finite strings of symbols, $D_1, D_2 \ldots D_n$. Given a new string, D_{n+1} , what is the probability that it belongs to the set? Or — given a string, a , how must it be completed so it is most likely to be a member of the set? Or, given a string a and a set of possible completions, $[ab_j]$, what is the relative probability of each of these completions?

A common example of unordered set prediction occurs in ethnic and formal languages. We are given a set of examples of strings that are acceptable sentences. Given a new string, what is the probability that it is acceptable? A common solution technique is to devise a well fitting stochastic grammar for the known set of strings. Algorithmic probability gives a criterion for goodness of fit of such grammars [Hor 71, Sol 64b,pp.240-251].

The supervised categorization problem is very similar. We are given a vector of properties associated with each of a set of mushrooms. The final property tells if it is poisonous or not. Given the vector of properties of a new mushroom, except for the final vector component - what is the probability that it is poisonous?

In numerical statistical analysis, we have as data, a set of pairs of numbers $[x_i, y_i]$. Given a new x_j, what is the probability distribution over all possible values of y_j?

We have an unordered set of ordered pairs of strings. Each pair represents a question and an associated correct answer. Given a new question string, find the most probably correct answer, or find the relative likelihood of several possible answers.

The problems of the last two paragraphs may also be thought of as examples of "stochastic operator induction". In the first paragraph, we want a stochastic function so that when we give it x_j as input, it will give the probability distribution of y_j as its output. Similarly, in the second paragraph, when we insert our "question" into the stochastic operator, we want a probability distribution on "answers."

Section 2.1 gives a formal solution to the sequential prediction problem in terms of a kind of universal Turing machine. Any describable probability distribution on strings can be represented by a suitable (not necessarily universal) Turing machine.

After some data has been analyzed, its statistical characteristics can be summarized by an apriori probability distribution on all possible subsequent data. Section 2.2 shows how to construct a Turing machine that represents such a distribution for sequential data.

Section 3.1 gives a formal solution to the problem of prediction of unordered data sets, also using a universal Turing machine.

Section 3.2 shows how to construct a Turing machine that summarizes an unordered data set and yields an apriori probability distribution for the next finite string.

Section 3.3 gives a different formal solution to the prediction problem for unordered strings that is closer to the approximations that are commonly used in solving such problems.

2   Sequentially Ordered Data

2.1 Algorithmic probability extrapolates the data string, D , by considering $P_M(E)$, the aprior probability distribution on all finite strings, E, with respect to M , a universal Turing machine.[1] $P_M(E)$ is the probability that the machine's output will be a string having E as prefix, if M's input is a completely random binary string.

From this definition, it can be shown [Sol 64a p.14 section 3.2, Sol 78, p.423] that

$$P_M(E) = \sum_k 2^{-|I_k|}$$

$[I_k]$ is the set of inputs to M such that M's output string has E as prefix. However the $[I_k]$ have the additional constraint that they are "minimal" in the sense that if the final bit of a I_k is removed, creating I'_k, then E is no longer a prefix of $M(I'_k)$. The $[I_k]$ form a prefix set. $|\,I_k\,|$ is the length of string I_k .

Suppose we have a known data string, D and we want to know the relative probabilities of various possible continuations of it. From Bayes' theorem, $P_M(DD_1)/P_M(D)$ is the (unnormalized) probability that if string D occurs, D_1 will follow.

It should be noted that the expression for P_M is incomputable [Sol 78, p.423]. For any practical applications we us approximations. A common method is to use only one term in the sum – that corresponding to the shortest code that we have been able to find for the data. This approximation is called Minimum Message Length [Wal 68] or Minimum Description Length [Ris 78]. If we use more terms in the sum that correspond to short codes, we get better approximations.

2.2 Question: Does there exist a machine M that summarizes the data D , so that $P_{M'}(D_1) = cP_M(DD_1)/P_M(D)$ Here, c is a constant that is the same for all possible D_1.

Observing the creation of D from the random input to M, one might think that such a machine could be obtained by stopping M as soon as it had produced D. Whatever internal states it had as well as the content of its memory tape at that time would then be regarded as the initial state of machine M.

An objection to this approach is that the resultant M' would characterize the state of M for *only one input* that would yield D (plus possible suffix) as output. It would neglect all the other possible codes of D and so would usually not be very close to $cP_M(DD_1)/P_M(D)$

---

[1]All machines herein discussed, whether universal or not, will have unidirectional input and output tapes, with one bidirectional memory tape.

A better method of constructing M':

M' is a box with four things in it. (1) M (2) a switch (3) a little man (4) a copy of D.

M'
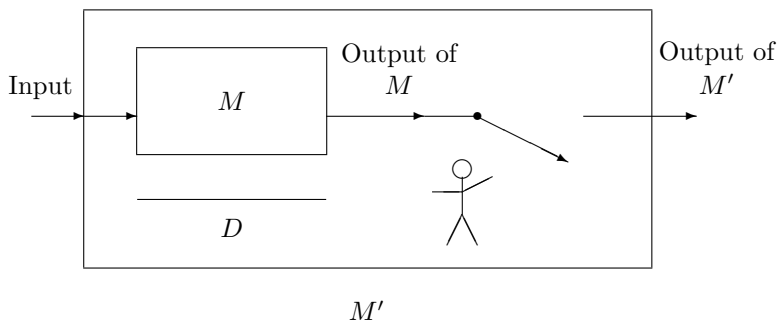
The (random) input to M' goes directly into M . The little man watches the output of M . If it matches D exactly, he switches the subsequent output of M to be the output of M'. If the initial output of M doesn't match D exactly, the M' has no output.

While $P'_M(D_1) = \mathrm{cP_M(DD_1)/P_M(D)}$ exactly, M' is not a very practical way to realize $P_M(DD_1)$, because *very* few of the inputs to M' produce any output. However, at this point we are only concerned here with theoretical models. In all practical cases, we use approximations. One of the simplest is suggested by the discussion of approximation of the previous section. We use as M', the state of M after it has produced D, using the shortest code for D that we have been able to find. It is not difficult to improve this model further by including other short codes for D.

3   An Unordered Set of Finite Strings

3.1 I will describe two different ways to deal with unordered data. Though they are equivalent, they shed light on different aspects of the problem.

The first way considers all possible finite bags of finite strings, $[D_k]$. D_k is the kth finite string. $k = 1, 2, \ldots, n$. A "bag" is similar to an unordered set of objects, but each type of object may occur more than once. We want a universal probability distribution, $P_M([D_k])$, over these bags.

$$P_M([D_k]) = \sum_j 2^{-|I_j|}$$

$| I_j |$ is the length of the jth description of the bag, $[D_k]$.

We can represent the set of strings, $[D_k]$ by the single string $A_1 = D_1@D_2@ \ldots @D_n$. The "@" are punctuation symbols. One way to describe $[D_k]$ is to write a self-delimiting program for string A_1 — a program that generates A_1 , then stops.

This amounts to a description of $[D_k]$ because from A_1, one can always find the *unordered* set, $[D_k]$. If the D_k are all different, there are $n!$ ways to order the D_k and $n!$ ways we can write A_j 's that describe $[D_k]$. If we define

$$P_M(A_1) = \sum_j 2^{-|I_j|}$$

in which $[I_j]$ is the set of all self delimiting programs for A_1, then

$$P_M([D_k]) = \sum_{j=1}^{n!} P_M(A_j)$$

If some of the D_k are identical, there are fewer different permutations and the sum will be over fewer A_j.

This definition of P_M can be used to give a formal solution to any of the prediction problems that were mentioned.

Suppose that $[D_k]$, $k = 1, \ldots, n$ is a sample of sentences generated by some unknown stochastic language. We are given a new string D_{n+1} , and we are asked the probability that it would be generated by the language.

A solution is $P_M([D_k] \cup D_{n+1})/P_M([D_k])$. Here $([D_k] \cup D_{n+1})$ is the $n + 1$ member bag that consists of $[D_k]$ plus the new member D_{n+1}.

3.2 The "Summarizing Machine" question corresponding to the problem in section 2.2 is:

Given a bag $[D_k], k = 1, \ldots, n$ and a string, D_{n+1}, can we devise a machine, M' such that

$$P_{M'}(D_{n+1}) = \mathrm{c} P_M([D_k] \cup D_{n+1}/P_M([D_k])$$

We construct the machine M' as before, with the following modifications:

The little man watches the output of M . If and only if it produces a string that is a suitably punctuated permutation of the $[D_k]$, he switches the output of M to be the output of M'. When M' (and M) subsequently have random input, then the probability that M' will produce D_{n+1} and either stop or follow with @ is proportional to $P_M([D_k] \cup D_{n+1}/P_M([D_k])$.

3.3 We shall describe a second method of dealing with unordered data that is closer in form to approximations that are most often used. First, let us consider the technique by which Minimum Message Length or Minimum Description Length deal with unordered data.

Let $[M_k]$ be a set of machines or algorithms. Each M_k is able to assign a probability to every conceivable finite string, D_j . Call this $P_{M_k}(D_j)$.

The probability assigned to the set $[D_j], j = 1, \ldots, h$ is

$$P(M_k) \prod_{j=1}^{h} P_{M_k}(D_j) \tag{1}$$

$P(M_k)$ is the probability assigned to M_k via its Minimum Message or Description Length, and M_k is chosen so that (1) is maximum. In general, maximum probability corresponds to minimum code length.

The algorithmic probability version is about the same, but it sums over all M_k's and obtains the $P_{M_k}(D_j)$ values a little differently.

Let $M_u(a, b)$ be a universal machine with 2 inputs.

Its first argument describes the machine it is simulating, and the second argument describes the input to the simulated machine. Since M_u is universal, for any machine M, there exists an a_M such that for all s,

$$M_u(a_M, s) = M(s) \tag{2}$$

The way this works: we put tape a_M into M_u. M_u runs until it has read all of a_M, and eventually it begins to ask for bits of s. At this point, it acts exactly like $M(s)$.

The universal algorithmic probability distribution is then:

$$P_{M_u}([D_n]) = \sum_j (2^{-|a_j|} \prod_{n=1}^{h} P_j(D_n)) \tag{3}$$

$2^{|-a_j|}$ is the probability associated with the jth machine description, a_j.

$P_j(D_n)$ is the probability assigned to D_n by the machine described by a_j.

$$P_j(D_n) = \sum_k 2^{-|r_{njk}|} \qquad r_{njk} \mid M_u(a_j, r_{njk}) = D_n$$

$\mid r_{njk} \mid$ is the length of the kth self delimiting program for D_n via $M_u(a_j, \cdot)$, the machine described by a_j.

Suppose $P(D_n)$ is any computable probability distribution over all finite strings, D_n. Then $P()$ can always be represented by a machine M_j , such that [2]

$$P(D_n) = \sum_k 2^{-|r_{njk}|} \qquad r_{njk} \mid M_j(r_{njk}) = D_n \tag{4}$$

" $M_j(r_{njk}) = D_n$ " means that for input r_{njk}, M_j prints D_n, then stops.

If a_j is a description of $P(\cdot)$, in that $M_u(a_j, \cdot) = M_j(\cdot)$, then $P_{M_u}([D_j])$ (equation (3)) includes in its summation, the term

$$2^{-|a_j|} \prod_{n=1}^{h} P_j(D_n)$$

Since $P_j(\cdot) = P(\cdot)$ for this value of a_j, it is clear that

$$P_{M_u}([D_n]) > 2^{-|a_j|} P([D_n]) \tag{5}$$

a_j can be any code for a machine representing P. If we select the shortest code, (5) will be a stronger statement. More generally, (5) says that

---

[2]The construction of such a machine is described in [Wil 70, p.252, Theorem 12], but see [Leu 78, Lemma of last theorem] for a more transparent demonstration.

$$P_{M_u}([D_n]) > cP([D_n]) \qquad\qquad (6)$$

and c is independent of $[D_n]$.

We can strengthen (6) further with a larger value of c by allowing c to be generated by the sum of all codes for M_j's that represent P: i.e.

$$c = \sum_j 2^{-|a_j|} \qquad a_j \mid \forall r\, M_u(a_j, r) = M_j(r) \qquad\qquad (7)$$

Here, $[M_j]$ is the set of all machines, M_j , that satisfy eq. (4).

Why are we interested in larger values of c? In sequential prediction problems, it has been shown that if we use $P_M()$ to estimate conditional probabilities of symbols in the data instead of $P()$, the probability distribution that actually generated the data, then our expected total squared error in probability estimates will be less than $-\frac{1}{2}lnc$ [Sol 78 p. 426, section IV].

The larger c is, the smaller our expected error. It is likely that a similar error bound will be found for unordered data, so, in this case as well, we want c as large as possible.

In some applications such as finance, probability estimates are competitive and small differences in error translate into sizable economic sums.

Large c means that the $[a_j]$ of equ(7) are small – that the machines, M_j that represent P, can be simulated by M_u using short codes .

ACKNOWLEDGMENT

References

[1] (Hor 71) Horning, J. "A Procedure for Grammatical Inference," *Proceedings of the IFIP Congress 71*, Amsterdam, North Holland, pp. 519–523, 1971.

[2] (Leu 78) Leung-Yan-Cheong, S.K. and Cover, T.M. "Some Equivalences Between Shannon Entropy and Kolmogorov Complexity," *IEEE Transactions on Information Theory*, IT-24:331–339,1978.

[3] (Li 93) Li, M. and Vitányi, P. *An Introduction to Kolmogorov Complexity and Its Applications*, Springer-Verlag, N.Y., 1993.

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

[5] (Ris 89) Rissanen, J. *Stochastical Complexity and Statistical Inquiry*, World Scientific Publishing Company, 1989.

[6] (Sol 64a) Solomonoff, R.J. "A Formal Theory of Inductive Inference," *Information and Control*, Part I: Vol 7, No. 1, pp. 1–22, March 1964.

[7] (Sol 64b) Solomonoff, R.J. "A Formal Theory of Inductive Inference," *Information and Control*, Part II: Vol 7, No. 2, pp. 224–254, June 1964.

[8] (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.

[9] (Wal 68) Wallace, C.S and Boulton, D.M. "An Information Measure for Classification," *Computing Journal*, 11:185–195, 1968.

[10] (Wil 70) Willis, D.G. "Computational Complexity and Probability Constructions," *Journal of the Assoc. of Comp. Mach.*, pp. 241–259, April 1970.