AFOSR 2214

ZTB-141

$$\left(\frac{f_A \cdot f_B}{f_{AB}} - 1\right)^2$$

AN INDUCTIVE INFERENCE CODE EMPLOYING DEFINITIONS

R. J. Solomonoff

APRIL 1962

CONTRACT AF 49(638)-376

PREPARED FOR

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
AIR RESEARCH AND DEVELOPMENT COMMAND
UNITED STATES AIR FORCE

WASHINGTON 25, D. C.

ZATOR COMPANY

140½ MOUNT AUBURN STREET, CAMBRIDGE 38, MASS.

AFOSR 2214

ZTB-141

$$\left(\frac{f_A \cdot f_B}{f_{AB}} - 1\right)^2$$

AN INDUCTIVE INFERENCE CODE EMPLOYING DEFINITIONS

R. J. Solomonoff

APRIL 1962

CONTRACT AF 49(638)-376

PREPARED FOR

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
AIR RESEARCH AND DEVELOPMENT COMMAND
UNITED STATES AIR FORCE

WASHINGTON 25, D. C.

ZATOR COMPANY
140½ MOUNT AUBURN STREET, CAMBRIDGE 38, MASS.

AN INDUCTIVE INFERENCE CODE EMPLOYING DEFINITIONS

R. J. Solomonoff

ABSTRACT

A new inductive inference method has been described in which the a priori probability of a sequence of symbols is computed on the basis of the lengths of various code strings that could be used to describe that sequence to a universal Turing machine.

A coding method for a string of symbols that form a Bernoulli sequence has been described in some detail. The present paper describes a code that is to be used when there are intersymbol constraints. This coding method defines certain special code symbols to be equivalent to certain substrings of symbols in the original sequence. The original sequence is then more economically coded using the special symbols. Both the cost of defining a special symbol, as well as the economy to be gained in using it, are considered in the decisions of what definitions to make.

To estimate the problems of implementation, a computer program was written to apply this technique to analysis and prediction in natural languages. Other applications are also discussed.

TABLE OF CONTENTS

ZTB
41

AN INDUCTIVE INFERENCE CODE EMPLOYING DEFINITIONS

R. J. Solomonoff

I. INTRODUCTION AND SUMMARY

A very general inductive inference method has been described (References 1 and 2) in which an a priori probability may be assigned to any long sequence of symbols.

The method consists of coding the sequence in a type of binary code, in all possible ways. The probability of any particular code is then 2^{-N}, N being the number of bits in its binary representation. The a priori probability of the sequence is then the sum of the probabilities of all of its codes, suitably modified by considering possible future continuations of the sequence, as well as factors to insure convergence.

The conditional probabilities of various possible individual symbols following a given sequence of symbols may then be obtained by taking the normalized a priori probabilities of sequences that consist of the given sequence, to which the various possible symbols have been concatenated.

In Reference 3, a coding method was described for a Bernoulli sequence – a sequence in which there were no probabilistic intersymbol constraints.

The present paper deals with sequences of symbols such as Markov chains, in which the probability of a particular symbol occurring at a particular point depends on the nature of symbols in the sequence that are close to it. If the entire sequence is very short, only very local interactions are

TB
41

considered. For longer sequences, more distant interactions are automatically considered.

Basically the coding method consists of defining certain sequences of two symbols — such as ab or db or ae — to be represented by special single symbols, such as α, β or γ. Using these newly defined symbols, the original sequence can be rewritten more compactly. However, the gain in compactness may be offset by the amount of information needed to write the definitions of the sequences defined.

The coding method of inductive inference gives a unifed measure to the "increase in compactness" brought about by the introduction of a particular definition, and includes the cost of defining the new symbol used.

The method to be described begins by considering all possible pairs of symbols. The pair for which the total decrease in "coding cost" is maximum is then assigned a special symbol, and the original sequence is rewritten using this special symbol. At the next stage all pairs of symbols (including the newly defined special symbol) are examined, and the pair for which decrease in coding cost is maximum is assigned a new special symbol. The sequence that is the result of the last rewriting is again rewritten using the new symbol.

This process is repeated again and again until it is no longer possible to find a new definition that results in a further reduction of "coding cost."

From the compact code that results we are able to find the a priori probability of the original sequence. This a priori probability can then be used to find the conditional probability of a symbol occurring at a particular point, in view of the nature of the symbols occurring before it in the sequence.

Section II describes an "intermediate code" in which new symbols are defined. Each of these new symbols represents a sub-string of symbols of the original sequence.

Section III shows how a priori probabilities are to be assigned to these intermediate codes.

Section IV discusses the use of these codes for computing approximate probabilities. A "hill-climbing" method is described by which codes of high a priori probability may be found.

Section V discusses several approximation formulae for the increase in a priori probability associated with each possible step on the "hill."

Section VI discusses the characteristics of the computer program that was written to implement the hill climbing routine of Section IV.

Section VII gives several possible applications of the computer program.

II. AN INTERMEDIATE CODE EMPLOYING DEFINITIONS

In the present paper we shall consider only definitions that involve the concatenation of two symbols. Since either or both of the symbols may in turn represent a concatenation of two symbols it is clear that we can in this way define sequences containing any desired number of symbols of the type used in the original uncoded sequence.

Suppose we have the sequence

$$\mathrm{CABCBABBABABAAB}. \tag{1}$$

Clearly, if we define the symbol α to represent the sub-sequence AB we can write (1) more compactly as

$$\mathrm{C}\alpha\mathrm{CB}\alpha\mathrm{B}\alpha\alpha\mathrm{A}\alpha. \tag{2}$$

However, in order to include all the information in our intermediate code, we must include the definition in our description of (1). A more complete code would then be

$$\mathrm{AB},\ \mathrm{C}\alpha\mathrm{CB}\alpha\mathrm{B}\alpha\alpha\mathrm{A}\alpha. \tag{3}$$

Here the comma is a special symbol. It occurs in every intermediate code once and only once. There are an even number of symbols before the comma, and adjacent pairs of these symbols are the definitions of the respective Greek letters.

The intermediate code

$$\mathrm{AB}\alpha\mathrm{A},\ \mathrm{C}\beta\mathrm{A}\alpha\alpha\beta\mathrm{CAA}\beta \tag{4}$$

would represent the fact that α is defined to be AB, and β is defined to be αA in the sequence

$$\mathrm{C}\beta\mathrm{A}\alpha\alpha\beta\mathrm{CAA}\beta.$$

It is clear then, that the sequence represented by (4) is

$$\text{CABAAABABABACAAABA.} \qquad (5)$$

III. ASSIGNMENT OF A PRIORI PROBABILITIES TO INTERMEDIATE CODES

To obtain the a priori probability of the sequence (5) we will represent its intermediate code (4) by a (usually large) positive integer in a manner that is basically similar to the coding method described in Reference 3.

Let us first number the symbols of (4) so that we may more easily discuss them.

$$\begin{array}{cccccccccccccccc} ABC, & A & B & \alpha & A & , & C & \beta & A & \alpha & \alpha & \beta & C & A & A & \beta \\ & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{array} \tag{6}$$

The symbols "ABC," have been written before the sequence proper as we have done in Reference 3.

In coding the first symbol of (6), we assign an a priori probability of $\frac{1}{4}$ to the symbol A. This is because there are 4 previous legal possibilities for that symbol, and only one of them is "A". The symbol "," in this position would indicate that no definitions were to be made in this intermediate code.

In coding the second symbol,"," is not a possibility since there must be an even number of symbols in the "definitions" section of our code. The legal symbol choices occurring before the second symbol are four in number: A, B, C and A. Of these only one is B so we assign an a priori probability of $\frac{1}{4}$ to B.

Since our first definition is now completed – the definition of α – we have a new symbol that is now possible. So (6) must be rewritten as

$$\begin{array}{cccccccccccccccc} \alpha ABC, & A & B & \alpha & A & , & C & \beta & A & \alpha & \alpha & B & C & A & A & \beta. \\ & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{array} \tag{7}$$

In coding the third symbol, "," and "α" are both legal, so we have seven legal possibilities, of which only one is α, so we obtain a priori probability $\frac{1}{7}$.

To code the fourth symbol we have seven legal possibilities (since "," is not legal here). Of these, two are A's, so we obtain a priori probability $\frac{2}{7}$.

In coding the fifth symbol we must rewrite (7) since we have completed the definition of β, and so β is now a possible choice.

$$\begin{array}{cccccccccccccccccc} \beta & \alpha & A & B & C, & A & B & \alpha & A, & C & \beta & A & \alpha & \alpha & \beta & C & A & A & \beta. \\ & & & & & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{array} \quad (8)$$

For the fifth symbol there are just ten legal previous possibilities. Only one of them is "," so its probability is $\frac{1}{10}$.

For the sixth symbol, and all subsequent symbols, "," is no longer legal since it can occur only once in the code. Therefore, the probability for the sixth symbol is $\frac{1}{9}$.

The probabilities of the seventh and eighth symbols are obtained straightforwardly – they are $\frac{1}{10}$ and $\frac{3}{11}$, respectively.

The ninth symbol brings up an interesting and very important point. If we have made the definition $\alpha \equiv AB$, then in our subsequent code, the symbol B should <u>never</u> follow A, since it would be more economical to rewrite the pair as α. In general, every definition that we make imposes some constraints on which symbols may follow which in the subsequent code.

In the present case, the ninth symbol cannot be B or ",". The resultant probability is then $\frac{2}{10}$.

The tenth symbol cannot be A since it follows α, and $\beta \equiv \alpha A$ has been defined. The probability assigned to the tenth symbol is therefore $\frac{3}{9}$.

The coding of the subsequent symbols illustrates no new points and will not be discussed further.

The final a priori probability of the sequence up to and including the tenth symbol is the product of the individual probabilities that were described, i.e.,

$$\frac{1}{4}\cdot\frac{1}{4}\cdot\frac{1}{7}\cdot\frac{2}{7}\cdot\frac{1}{10}\cdot\frac{1}{9}\cdot\frac{1}{10}\cdot\frac{3}{11}\cdot\frac{2}{10}\cdot\frac{3}{9}$$

IV. USE OF THE CODES FOR PREDICTION

Suppose we have a string of symbols which we will denote by T and we want to know the relative probability of the symbol a, rather than b, being the symbol that follows T.

Equation 5 of Reference 1 computes this probability ratio by considering all codings of all possible continuations of the sequences Ta and Tb. In general, a given sequence can be coded in many ways. Various sets of definitions can be used, and they can be defined in different orders – e.g., AB can be defined before BC, or vice versa. Also, with a fixed set of definitions, it is often possible to code a sequence in several different ways. As an example, suppose we have defined $\alpha \equiv AB$ and $\beta \equiv BC$. Then the sequence ABC can be coded as either αC or $A\beta$.

An approximation to the desired probability ratio can be obtained by considering only a few codings of only a few possible continuations of these two sequences. Greater accuracy will, of course, be obtained if more codings and more possible continuations are considered, and if the coding methods used are of relatively high a priori probability.

A computer program has been written for prediction in which only the sequences Ta and Tb are coded. Possible future continuations are not considered. An attempt is made to find codes of very high a priori probability by a process of "hill climbing" – i.e., the original sequence is used as an initial code and improvements are made in this code so as to increase the a priori probability. This results in a new code which is in turn improved. This process of improvement continues until no new improvements can be found within the set of improvement types being considered.

In the particular computer program that was used, each "improvement" consists of devising a new binary definition and rewriting the previous intermediate code using this new definition. If there are d symbol types in the original sequence, and c definitions have been introduced thus far, then $(d + c)^2$ possible definitions are considered. The definition that results in the largest increase of a priori probability of the intermediate code is used for recoding. Next, $(d + c + 1)^2$ definitions are considered and the optimum one is selected.

The process of determining how much the introduction of a given definition will increase the a priori probability of a code, has been studied at length. Several approximate formulae have been obtained for the resultant change in a priori probability. The approximations are easily implemented by a digital computer, so that with an initial symbol string of about 1000 symbols it takes only a few minutes for the IBM 709 computer to find as many as 100 new optimum definitions.

V. APPROXIMATION FORMULAE FOR HILL CLIMBING

In general, there are two situations in which it is useful to define the ordered pair, AB.

In the first kind of situation, we have a long uncoded sequence in which the subsequence AB never (or very infrequently) occurs. Defining the pair AB will then increase the a priori probability of the resultant intermediate code. This is because of increased knowledge of symbols following A's due to the fact the B's are impossible there. This knowledge results in greater probabilities being assigned to the symbols that actually do follow A in the intermediate code.

In the other possible situation, B follows A almost whenever A occurs. We can then increase the a priori probability of our intermediate code by defining AB.

If f_A and f_B are the respective relative frequencies with which the symbols A and B occur in a long, uncoded sequence, and f_{AB} is the relative frequency of the adjacent pair, AB (relative frequency is in all three cases measured with respect to the total number of single symbols in the sequence), then the ratio of total increase in a priori probability of the intermediate code resulting from our formulating the definition $\alpha \equiv AB$, is roughly approximated by

$$f_A f_B \exp\left(\frac{kf_{AB}}{2}\left(\frac{f_A \cdot f_B}{f_{AB}} - 1\right)^2\right) \tag{9}$$

Here, k is the number of symbols in the original sequence and so kf_{AB} is the number of times AB has occurred.

We will want to define $\alpha \equiv AB$, if this definition gives us an intermediate code of higher a priori probability – i.e., if expression (9) is greater than unity.

It is of value to regard expression (9) as composed of two factors. First there is the factor $f_A \cdot f_B$, which is the cost (in probability) of writing the definition AB. Next, the factor

$$e^{\frac{kf_{AB}}{2}\left(\frac{f_A \cdot f_B}{f_{AB}} - 1\right)^2} \qquad (10)$$

which tells us how much benefit is obtained from the definition's increasing the probabilities of various symbols.

In (10) note the presence of the expression

$$\left(\frac{f_A \cdot f_B}{f_{AB}} - 1\right)^2$$

This indicates that if there is any <u>constraint</u> between the symbols f_A and f_B, so that $f_{AB} \neq f_A \cdot f_B$ (i.e., A and B are not "independent"), then, if our sequence is long enough (i.e., k is large enough), expression (10) will become very large — so that we will save more than we lose by defining $\alpha \equiv AB$.

We may write (10) as

$$\left[e^{\frac{f_{AB}}{2}\left(\frac{f_A \cdot f_B}{f_{AB}} - 1\right)^2}\right]^k$$

Here it becomes clear that no matter how much A and B appear to be dependent (i. e., that f_{AB} differs very much from $f_A \cdot f_B$), it will not be worth while to define $\alpha \equiv AB$ unless the sequence is long enough to give us an adequate "sample size" (i.e., k is large enough). Conversely, even if A and B are but very slightly dependent, it will be worth while to define $\alpha \equiv AB$ if k is large enough.

Also note that if A and B are rather uncommon symbols (i.e., f_A and f_B are small) the cost of defining AB is very important (i.e., $f_A \cdot f_B$ is much smaller than unity), so that the other factor, (10), has more "work" to do. This corresponds to the general epistemological principle that if a regularity is a priori very unlikely, we require much more empirical evidence to convince us that it is indeed a bona fide "regularity."

In general, in the "coding method of inductive inference," it will often be possible to divide the coding effects of a supposed "regularity" in a body of data into a part that corresponds to the cost of defining the regularity, and a part that tells how much we increase the a priori probability of the code by using this regularity in recoding the data. In equation (3) these parts are exemplified by $f_A f_B$ and by expression (10), respectively.

It should be noted that the approximation expression (9) does not work well for very small values of f_{AB}, and no conclusions should be drawn about this case from this particular approximation.

Expressions (11) and (12) are more exact expressions for the ratio of increase of a priori probability of an intermediate code that results from defining $\alpha \equiv AB$. These expressions were the ones used in the computer program. Expression (11) is for the case $A \not\equiv B$, and (12) is for the case $A \equiv B$ — i.e., $\alpha \equiv AA$. Although both expressions are approximations, they work very well, even when $N_{AB} = 0$ or $N_{AA} = 0$.

$$\frac{(N_A-N_{AB}+1)!}{N_A!}\,\frac{(N_B-N_{AB}+1)!}{N_B!}\,\frac{(k+d+c-1)!\,N_{AB}!\,(d+3c)}{(k+d+c-N_{AB}+2)!}\left(\frac{k+d+c-N_{AB}+3}{k+d+c-N_B+1}\right)^{N_A-N_{AB}} \tag{11}$$

$$\frac{(N_A-2N_{AA}+2)!\,N_{AA}!}{N_A!}\,\frac{(k+d+c-1)!\,(d+3c)}{(k+d+c-N_{AA}+2)!}\left(\frac{k+d+c-N_{AA}+3}{k+d+c-N_A-1}\right)^{N_A-2N_{AA}} \tag{12}$$

k is the total number of symbols in the original sequence.

d is the number of different symbol types in the original sequence.

c is the number of definitions that have been introduced.

N_A is the number of times the "A" occurs in the code sequence (or original sequence) being recoded.

N_B is the corresponding number for "B".

N_{AB} is the corresponding number for the subsequence "AB".

N_{AA}. is the corresponding number for the subsequence "AA".

Here N_{AA} is defined to be the largest number of non-overlapping "AA's" that can be found in the sequence of interest.

In expression (11), the factor

$$\left(\frac{k+d+c-N_{AB}+3}{k+d+c-N_B+1}\right)^{N_A - N_{AB}} \qquad (13)$$

gives the amount of increase in a priori probability due to symbols following A having somewhat greater probability than before – since it is now known that B cannot follow A. This factor is but an approximation, and assumes that this increase in probability is the same for each of the $N_A - N_{AB}$ occurrences of "A" in the new code. The rest of expression (11) is exact however. Corresponding remarks are true of expression (12).

VI. THE COMPUTER PROGRAM

A computer program has been written in order to study the practical problems of implementing the "Hill Climbing" procedure described in Section IV, and to estimate expected computing times. The program has not been run. The input data consists of a sequence of alphabetic symbols. The output consists of (1) an ordered set of definitions of ordered symbol pairs, (2) the intermediate code of the original sequence, using these definitions, (3) the a priori probability of that code sequence.

We need only the third of these outputs to make probability estimates. Suppose that for the string of input symbols designated by a, the machine gives us a code whose a priori probability is M(a). If the symbols used in the original sequence are A, B and C, then an approximation to the probability that A will be the next symbol to follow the sequence a, is given by:

$$\frac{M(aA)}{M(aA) + M(aB) + M(aC)} \qquad (14)$$

Here, aA is the concatenation of a and A.

The program was written almost entirely in Fortran II for the IBM 709 or 7090. The only non-Fortran subroutines that were used were rather simple input and output subroutines. Extensive use was made of tables to implement formulae (11) and (12) as rapidly as possible.

Most time consuming operation of all was the comparison of the results of using each of the $(c + d)^2$ possible definitions, and selecting the "best" one. An enormous amount of time was saved by noting that for almost all AB pairs, $N_{AB} = 0$. Furthermore, both (11) and (12) are almost monotonic in N_A and N_B for $N_{AB} = 0$. This makes it possible to find an AB for which (11) is maximum over the set of AB for which $N_{AB} = 0$. This can be done without computing the value of (11) for many AB pairs. We need then use formula

(11) to find the best AB for which $N_{AB} \neq 0$ and compare it with the best AB for which $N_{AB} = 0$.

This particular trick works very well only if the original sequence is not very long. Specifically, if

$$k < (d + c)^2 \tag{15}$$

where k, c and d are the same as in expressions (11) and (12).

If (15) is not true, some slight modifications of the above technique may be desirable.

If (15) is true, then the computation time to climb c locally optimum steps up the hill (i.e., to find c new definitions) takes the IBM 709 about ck milliseconds plus about 5 minutes for compiling. For $k = 1000$ and $c = 100$, this amounts to a total of about 7 minutes of computer time. Using the 7090 reduces the time required to about one fifth of this figure.

If (15) is not true and the computation technique is unmodified, a corresponding time would be $c(c + d)^2$ milliseconds plus 5 minutes for compiling for the IBM 709 .

For problems in which k, c and d are much larger, there are various techniques that can be used to decrease the computation time. If $(c + d)^2$ is greater than the number of words available in the core memory, then the program must be modified somewhat. It is likely that $c + d = 1000$ can be dealt with by using a core memory of 32000 words without radical reprogramming.

VII. APPLICATIONS OF THE PROGRAM

Only a few of the more important applications will be listed.

First, and probably most important, the program will give an empirical check on the effectiveness of the coding method of inductive inference. One way to do this is to compare its prediction accuracy for English text, with prediction accuracy of a more conventional method, using known digram and trigram frequencies.

Another check would be to code a large body of English text and obtain a large set of definitions from this coding. Then code a new corpus of English text using the same set of definitions. The redundancy obtained from the two codings should be about the same. If they are, this will verify the estimates made of the a priori probabilities of various defined sub-sequences.

This routine could be used for a kind of literary or musical detective work. Suppose one has a body of text by an author whose identity one wishes to establish. It is known that the author is one of three people, and samples of the writings of each of these suspects is available. The text by the unknown author is coded and a set of definitions is obtained. This fixed set of definitions is then used to code the works of each of the suspects. The most likely suspect is then the one whose known writing has a redundancy matching most closely that of the text of the unknown author. The same sort of analysis is applicable to musical compositions.

"Monte Carlo music" can be written that is "similar" to music in a small sample of compositions analyzed. The present technique makes better use of the available data than do other methods for constructing "Monte Carlo music." This makes it possible to recognize more complex sequences in the data than would be warranted if ordinary n-gram frequencies were used in the conventional way.

Given a sequence of Morse Code dots and dashes without the spaces between letters indicated, it may be possible to use the present routine to discover the letter groupings, and some short word groupings. A similar problem is the discovery of roots, prefixes and suffixes from a large English text.

The prediction routine could be used to make a penny matching machine of perhaps greater effectiveness than any such existing machine.

To some extent, the present computer routine could learn to recognize the legality of various Roman numerals.

The advantage of the present method of prediction over conventional methods using the frequencies of n-grams of fixed length, is that the present method is able to propose rather complex definitions and evaluate their significance on the basis of a relatively smaller amount of data. Using the same amount of data, the new method <u>should</u> be able to make better predictions than more conventional methods.

VIII. ACKNOWLEDGEMENT

That the particular method of Hill Climbing used in the present program might be successful, was a belief that was strongly encouraged by the success of R. Silver (Reference 4) in an analogous Hill Climbing routine used to solve simple substitution ciphers.

Mr. Silver's aid was also invaluable in preparing the computer program herein described.

REFERENCES

1. R. J. Solomonoff, "A Preliminary Report on a General Theory of Inductive Inference," Zator Technical Bulletin No. 138, AFOSR TN-60-1459; Zator Company, November 1960. (Revision of Report V-131, February 1960.)

2. R. J. Solomonoff, "Progress Report: Research in Inductive Inference for the Period 1 April 1959 to November 1960," Zator Technical Bulletin No. 139, AFOSR 160; Zator Company, January 1961.

3. R. J. Solomonoff, "A Coding Method for Inductive Inference," Zator Technical Bulletin No. 140, AFOSR 510; Zator Company, April 1961.

4. R. Silver, "The Decryptor Program;" Bolt, Beranek and Newman, Cambridge, Mass., (forthcoming).