INDUCTIVE INFERENCE THEORY — A UNIFIED APPROACH TO PROBLEMS IN PATTERN RECOGNITION AND ARTIFICIAL INTELLIGENCE

Ray J. Solomonoff

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

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

Abstract

Recent results in induction theory are reviewed that demonstrate the general adequacy of the induction system of Solomonoff and Willis. Several problems in pattern recognition and A.I. are investigated through these methods. The theory is used to obtain the a priori probabilities that are necessary in the application of stochastic languages to pattern recognition. A simple, quantitative solution is presented for part of Winston's problem of learning structural descriptions from examples. In contrast to work in non-probabilistic prediction, the present methods give probability values that can be used with decision theory to make critical decisions.

Introduction

The kind of induction theory that we will consider may be regarded as a Bayesian method in which the a priori probability of a hypothesis is related to the shortest descriptions of that hypothesis that are obtainable by programming a reference universal Turing machine.

The probability values obtained are ordinarily not effectively computable. They can become effectively computable if we make certain reasonable restrictions on the source of the data, but in either case they do not appear to be

1

practically calculable. However, various approximation methods readily suggest themselves, and indeed, all known methods of obtaining probability estimates may be regarded as approximations to the idealized method and they can be compared and criticized on this basis.

Several problems in pattern recognition and A.I. will be discussed with respect to this general formulation of induction.

Induction and pattern recognition through stochastic grammar construction is discussed in a general way. The best work in this area involves a Bayesian analysis and the present induction theories tell how to obtain the necessary a priori probabilities of various grammars.

Next we discuss in some detail part of Winston's program for learning structural descriptions from examples. A simple vector model of the problem is described and various quantitative results are readily obtained. These agree for the most part, with Winston's qualitative heuristic discussions of the relative likelihoods of various models of concepts, but the results are obtained very directly and do not involve tree search of any kind.

In addition to resolving many otherwise ambiguous decisions, the quantitative probabilities obtained enable us to use decision theory to make critical decisions as in the mechanization of medical diagnosis.

1 Recent Work in Induction

We will present some recent results in the theory of inductive inference — discussing what is possible and what is not possible.

For the purposes of this discussion, the problem of inductive inference will be the extrapolation of a sequence of symbols emitted by an unknown stochastic source. It is not difficult to show that almost all, if not all problems usually regarded as induction, can be expressed in this form. Discovery of multidimensional patterns, curve fitting, time series extrapolation and weather prediction are but a few of the kinds of problems that can be readily dealt with.

Although induction has always been the most important thing going on in science and has constituted a large part of the study of the philosophy of science, there has not, until recently, been a rigorous formulation of the process with a clear understanding of the expected errors involved. We will discuss these recent results and what they imply about what is possible, impossible and approximatable.

Recent work in induction has centered about the concept of the "description" of a string of symbols. A "description" of a string with respect to a particular reference computer is an input to that computer that gives the described string as output. Solomonoff [8] used the lengths of short descriptions of a string and its possible continuations to define the a priori probability of that string. Bayes' Theorem was then used to find the probability of any particular continuation of the string. He also showed that using a universal Turing machine for reference made the a priori probabilities relatively insensitive to choice of reference computer.

Very approximately,

$$P = \sum_{i=1}^{\infty} 2^{-N_i}$$

Here, P is the probability of the described string and N_i is the length of the ith possible description. We sum over all possible descriptions of the string and all possible continuations of it. Clearly the shortest descriptions get most weight, but the other descriptions can't be ignored.

Later, Kolmogorov [1] and Chaitin [13] proposed that a random sequence be defined to be one whose shortest description with respect to a universal Turing machine is about the same length as the sequence itself. Martin Löf [2], Loveland [3] Schnorr [4] continued work on randomness definitions. For a review of this work as well as subsequent research in the Soviet Union, see Zvonkin and Levin. [5].

More recently, Chaitin [6] has proposed expressions for entropies of sequences based on descriptions that form a prefix set. The expressions, however, have error terms that do not occur in the earlier, more exact formulation of Willis [9].

Willis refined Solomonoff's model and overcame several difficulties in it. The theorems in the present paper usually follow directly from his work. Because of the "halting problem", it is often impossible to tell if one string is a description of another with respect to a specific machine. Willis dealt with this problem by considering an infinite sequence of machines, each more powerful than the last, but all of them sufficiently limited so that they have no "halting problem." Associated with each of these machines is a computable probability assigned to the string in question.

One sequence of such machines is obtainable by considering a universal 3 tape Turing machine with unidirectional input and output tapes and a bidirectional working tape. The Tth machine in the sequence is obtained by stopping the universal machine after T steps (if it has not already stopped). It is not difficult to show that the sequence of probabilities obtained by these machines approaches a limit as T approaches infinity, but that the limit is not effectively computable.

Suppose that $A^{(m)}$ is a string of length m, and that we have a stochastic generator that assigns a probability $P(A^{(m)})$ to $A^{(m)}$. Suppose this generator is describable by a finite string b bits in length. Then for sufficiently powerful reference machines,

$$P^{M_T}(A^{(m)}) \geq 2^{-b} P(A^{(m)})$$

Here P^{M_T} is the probability assigned to $A^{(m)}$ by the reference machine, M_T. From this, it is clear that the limit machine must satisfy this inequality, so if we define

$$P^M(A^{(m)}) \equiv \lim_{T \to \infty} P^{M_T}(A^{(m)})$$

3

Then,

$$P^M(A^{(m)}) \geq 2^{-b} P(A^{(m)})$$

Note that the factor 2^{-b} is independent of m, the length of the sequence. The error between P^M and P is

$$\frac{P^M(A^{(m)})}{P(A^{(m)})} \geq 2^{-b}$$

One interpretation of the result is given by the work of Cover [7], who shows that if one places bets at even odds on a stochastic sequence generated by P, but bases one's bets on P^M, then the ratio of one's fortune to the optimum obtainable (which would use P as basis) will be just P^M/P.

Cover also proposed a somewhat different expression for P^M from Willis', basing it upon Chaitin's [6] definition of prefix code complexity. He then showed that for his probability definition, $P^{M'}$,

$$\lim_{m \to \infty} \frac{1}{m} ln(P^{M'}(A^{(m)})/P(A^{(m)}) = 0$$

which is a somewhat weaker result than Willis'. We have been able to show, however, that for a very broad class of stochastic sources, Cover's probability estimate is negligibly worse than Willis'.

To obtain a different measure of the accuracy of Willis' method, let $A^{(n)}$ represent the string consisting of the first n symbols of $A^{(m)}$ Then

$$\delta_n \equiv \frac{P(A^{(n+1)})}{P(A^{(n)})}$$

and

$$\delta'_n \equiv \frac{P^M(A^{(n+1)})}{P^M(A^{(n)})}$$

are the conditional probabilities of the $n + 1^{th}$ symbol, given the previous symbols, and

$$P(A^{(0)}) \equiv P^M(A^{(0)}) \equiv 1$$

From these definitions we obtain directly,

$$\frac{P^M(A^{(m)})}{P(A^{(m)})} = \prod_{n=1}^{m} \frac{\delta'n}{\delta n} \geq 2^{-b}$$

This expression is the ratio of the products of the conditional probabilities for the nth symbols.

If we want to know the mean error in this ratio we take the mth root and obtain $2^{-\frac{b}{m}}$. It is clear that it must approach unity as m becomes very large.

Although this is a very powerful result and gives real assurance that δ'_m converges to δ_m very rapidly as m increases, it lacks a certain intuitive appeal. One asks whether it could not be possible that the ratio might be $\ll 1$ for some factors and $\gg 1$ for others. While the mean could be close to unity the individual deviations could be quite large.

This difficulty does not arise, however, and from the foregoing result it is possible to show that the expected value of the total squared error between the Conditional probabilities δ_n and δ'_n remains $\ll bln\sqrt{2}$

$$E(\sum_{i=1}^{m}(\delta'_i - \delta_i)^2) \equiv \sum_{k=1}^{2^m}(P(^kA^{(m)})\sum_{i=1}^{m}(^k\delta'_i - ^k\delta_i)^2) \leq bln\sqrt{2}$$

Here E is the expected value with respect to P. $^kA^{(m)}$ is the kth sequence of length m. There are just 2^m of them. $^k\delta'_i$ and $k\delta_i$ are the conditional probabilities for the ith bit of $^kA^{(m)}$ for P^M and P, respectively.

The expected value of the mean square error between the conditional probabilities is less than $\frac{b}{m}ln\sqrt{2}$.

A few objections suggest themselves. First, this error is unreasonably smaller than those obtained in ordinary statistical analysis. For a simple Bernoulli sequence, the expected squared error in the mth symbol is proportional to $\frac{1}{m}$ and the total squared error is of the order of $ln\,m$ rather than $bln\sqrt{2}$.

The reason for this discrepancy is that we have assumed that the stochastic source had a finite description. If the source consisted of zeros and ones with probabilities 3/8 and 5/8 respectively, we could, indeed have a finite description — perhaps of the order of 4 or 5 bits. 0rdinarily in statistics, however, the stochastic sources are describable as continuous functions of a finite number of parameters. The parameters themselves each have infinitely long descriptions.

For situations of this sort, the expected total squared error is not bounded, but is roughly proportional to the log of the sequence length as in conventional statistical analysis. The error itself, however, approaches zero.

If there are k different parameters in the model, the expected total squared error will be bounded by

$$bln\sqrt{2} + Akln\,m$$

Here, m is the number of symbols in the string being described, A is a constant that is characteristic of the accuracy of the model and b is the number of bits in the description of the expression containing the k parameters.

If we are comparing several different models and we have much data (i.e. large m), then the "b" terms will be of little significance and we need only compare the corresponding Ak values to determine the best model.

A technique very similar to this has been successfully used by Akaike [10] to determine the optimum number of parameters to use in linear regression analysis. The present methods are, however, usually of most interest when the amount of directly relevant data is relatively small.

Another objection is that P^M is incomputable. This makes it impossible to calculate it exactly. Is there not a better solution possible? We can obtain progressively more computable solutions by making more and more restrictions on the stochastic source.

If there are no restrictions at all, there is no prediction at all possible. If we restrict the stochastic source only to be finitely describable then it is well known that there can be no effectively computable solutions, though as we have seen, there is a non effectively computable solution.

If we restrict the "computational complexity" of the stochastic source, so that we have an upper bound on how long it takes the source to compute the probability of a sequence of length m, then the probabilities are, indeed computable, and they converge as rapidly as they do in incomputable cases. They are, however, no more practically computable than the incomputable solutions.

In any case, we must use approximation methods. It is clear that the incomputable method described converges very rapidly — it is likely that it has less error for a given amount of data than any other probability evaluation method, and so we will do well to try to approximate it.

In most cases, our approximations consist of finding not the shortest description of the string of interest (this, too, being incomputable), but rather the *set* of the shortest descriptions that we can find with the resources available to us. All of the methods ordinarily used to estimate probability by finding regularities in data can be expressed in the form of discoveries of short descriptions of the data and therefore are approximations to the ideal method. In this common format, various approximations can be compared, so we can select the best ones — the ones most likely to give good predictions.

2 Application to Pattern Recognition Stochastic Language

To illustrate the problem, suppose we are given a set of strings of symbols, such as aa, abba, aabbaa, baaabbaaab etc., and we are told that these strings were acceptable sentences in some simple formal language. We are required to find a grammar that could generate these strings.

In general, there will be an infinite number of grammars that can generate the set even if we restrict the grammars, say, to be finite state or to be context free grammars. Early investigators proposed that some criterion of "simplicity" be imposed on the grammar, and various explications of this concept were devised, but with no basis for preferring one explication over any other.

By assuming the set of strings was produced by some unknown stochastic generator, and using some of the previously described methods to give an a priori probability distribution over all such generators, a very general solution to this problem is obtained, and the concept of "simplicity" is unnecessary. A stochastic language is an assignment of probabilities to all strings being considered. A stochastic generator or grammar is a means for carrying out this assignment.

One very general form of stochastic grammar consists of a Turing machine. One inserts the string into the machine and it prints out the probability of that string. Many of the strings may have zero probability assigned to them.

Another very general form of stochastic grammar is a generative grammar. Again we have a Turing machine, but we insert a random string of zeros and ones. The output strings of this machine then have the probability distribution associated with the desired stochastic language.

To use stochastic languages to solve induction problems, such as the one of guessing the grammar that produced a given set of strings, we first assume an a priori distribution on all possible stochastic languages. If P_i is the a priori probability of the ith stochastic language, and L_{ij} is the probability that the ith language will produce the jth sample string (there being m sample strings), then the most likely grammar is the one for which

$$P_i \prod_{j=1}^{m} L_{ij}$$

is maximum.

Often we are not interested in knowing which grammar produced the set of strings, we only want to know the probability that a particular new string is in the set — this new string being the $m+1^{th}$. A Bayesian analysis gives us

$$\frac{\sum_i P_i \prod_{i=1}^{m+1} L_{ij}}{\sum_i P_i \prod_{j=1}^{m} L_{ij}}$$

for this probability, the summation being taken over all grammars for which $P_i > 0$.

The a priori probability of a language corresponds roughly to the earlier concept of simplicity, but it is a more clearly defined quantity.

To generate a stochastic grammar from a non- stochastic generative grammar is usually very easy. In the generative grammar, at each point in the construction of the final object, there will be choices to be made. If we assign probabilities to each of these choices, we have a stochastic generative grammar. The probability of any particular derivation will be the product of the probabilities of the choices involved in that derivation. If the language is ambiguous, some objects will be derivable in more than one way, and the probabilities of each of these derivations must be added to obtain the total probability of the final object.

Many different kinds of grammars have been devised for various problem areas [11]. In particular, they have been used for two dimensional scene analysis, recognition of handwritten characters, chromosome pattern recognition, recognition of spoken words, etc.

The basic model that stochastic grammars propose is a very attractive one. It assumes that the sample set was created by some sort of mechanism and that the mechanism had various probabilistic elements. It enables us to put into the

7

model any information that we have about the problem, either deterministic or probabilistic. For some time, however, the assignment of a priori probabilities to the different possible mechanisms was a problem of uncertain solution.

Induction theory made an important breakthrough by providing a general method to assign probabilities to these mechanisms, whether the assignment is to be purely a priori or whether they are dependent in any way on available information.

If the probability is purely a priori, one method of probability assignment proceeds by writing out a minimal description of the non-stochastic grammar from which the stochastic grammar is derived. The details of how this is done for a kind of finite state grammar and for a general context free grammar are given in Ref. 8, pp. 232–253.

If there is some data available on the relative frequencies with which the primitive concepts have been used in the past for other induction problems, this information can be used to assign initial "bit costs" to these concepts when constructing new grammars.

3   Application to A.I.
Learning Structural Descriptions from Examples

Winston's program for learning various structural concepts from both positive and carefully chosen negative examples of those concepts [12] is perhaps the most competent induction program yet completed.

The program does much more than learn concepts, but we shall discuss only this particular part of the program. It begins with a line drawing of three dimensional objects. This consists of one or more objects in various positions, having various relations with respect to one another. There is a preprocessing program that translates this drawing into a description that is in the form of a net.

The nodes of the net are objects in the drawing, properties of objects and classes of objects. There are arrows connecting the nodes that denote relations between them.

For an example, suppose the original drawing pictured a cube on the left and a vertical brick on the right. A possible net describing this scene would have four nodes: a brick; a cube; the property, "standing" and the class of solids, "prism". The brick and cube are connected by an arrow labeled "to the right of". The brick has an arrow labeled "has the property of" connecting it to "standing". Both the brick and cube have arrows going to prism labeled "a-kind-of", indicating class inclusion.

After being shown several scenes that are given as positive, and several as negative examples of a certain concept, such as a table, the program tries to induce the concept by making a model of it. These models consist of networks similar to those used to describe scenes, but the nodes and arrows of the net are

8

usually *classes* of objects, classes of properties and classes of relations. These classes may sometimes be expressed as negations, e.g. "not a brick", or "is not to the left of".

For purposes of comparing scenes to one another and scenes to models, we will consider a scene description to be an ordered sequence of objects and relations — a vector whose components are a mixture of objects and relations. A "Model" is a vector whose components are classes.[1] Formalized in this way, it is clear that given a set of positive and negative examples, there will ordinarily be an enormous number of models such that

1. In each of the positive examples, all of the components are members of the corresponding classes in the model.

2. In each of the negative examples, at least one component is not a member of the corresponding class in the model.

Winston has devised a system for ordering the models, so that after each example is given, it picks the "best" model that is consistent with the data thus far — i.e. highest in the ordering. His ordering of models is very close to that of a priori probabilities as calculated by induction theory.

In one case we are given as a positive example, a brick that is standing on one end. The negative example is a brick lying on one side. The classes that are considered for the "property" component of the vector of the model are:

1. Standing

2. Not lying

Winston notes that since most concepts are defined in terms of properties rather than anti-properties, "standing" is more likely. In the language of induction theory, positive concepts that have names in the language are usually of high a priori probability. A negative concept consists of a positive concept with a negation symbol — which increases description length and decreases a priori probability. If a negative concept is of much utility in description, and of high a priori probability, it will have been useful to define a special word for it, e.g. "dirty" is the word for "not clean." Reference 8, pp. 232–240 treats the problem of when it is worth while to define new symbols.

Another reason why "standing" is a better choice than "not lying" is that "standing" probably has fewer members. This is brought out more clearly in the next example. Here we have two positive cases: 1) Apple 2) Orange. Three classes for the model are considered. First, the Boolean sum class of apples and oranges. Second, the class "fruit". Third, the universal class. Though this is not the example Winston gives, the choice he would make would be "fruit". He regards this as a sort of middle-of-the-road stand in a difficult induction problem.

---

[1]This differs slightly from Winston's definition of "model", but should cause no great difficulty.

9

Induction theory gives a quantitative discussion. To make this particular problem non-trivial, we assume that the positive examples given are in some sense "typical". If they are not constrained in this way, then the Universal class is the best response. Let P_i and N_i be the respective a priori probability of a class and the number of members in that class. Then if the positive examples are "typical" or, more narrowly, if all positive cases of the concept have equal likelihood of being given as examples, then by Bayes, the most likely class is the one for which P_i/N_i^n is maximum, n being the number of positive cases — two in the present situation.

The universal class is of high a priori probability, but it has very many members. The Boolean sum class has only two members, but its a priori likelihood tends to be low. This is because the symbol for Boolean sum is "expensive" — i.e. concepts formed using it, tend to be rather ad-hoc and not very useful in prediction. If the class "fruit" is, indeed, of fair a priori probability and there aren't too many kinds of fruit, it may well be the best choice. We might also consider "edible fruit" or "plants" depending on the sizes and a priori probabilities of these classes.

From these and similar heuristic arguments, Winston is able to order the possible models with respect to likelihood. The result is a remarkably capable program in the problem area he has selected. However, even in this limited area it is easy to get into complexities that are well beyond the capacity of the program. Winston deals with these by arbitrarily cutting off the consideration of certain possibility branches.

With expansion of the language to deal with a larger universe of problems — e.g. the inclusion of facilities for making recursive definitions — the necessary complexity would rapidly get beyond the capabilities of the heuristic discussions of likelihood that Winston uses.

We will describe a system of learning concepts that is similar to Winston's, but obtains quantitative probabilities and appears to be far simpler to implement. It is hoped that this simplification will make it possible to extend Winston's methods to much richer worlds of problems.

The system consists of a procedure for obtaining an initial model and for modifying the model as each new positive or negative example is given. At each point in time several of the best models are stored and these can be used to give the probability of any particular object being an example of the concept being learned.

We start out with a positive example of the concept.

Our first model has for its components, those classes of which the corresponding components of the example are members, and for which P_i/N_i is maximum, i being the component considered. Often these classes will have but one component and Winston uses classes of this kind for his initial model.

Subsequently, there are four possible example situations with respect to the model. A positive example can be either accepted or rejected by the model or a negative example can be accepted or rejected by the model. We will treat these one by one.

If a negative example is rejected by the model, we leave the model invariant.

Our criterion of choice of class is maximum P_i/N_i^n. Since n is the number of positive examples thus far, the new negative example does not modify this quantity in any way for any class, so a class that was optimal before the example must be optimal after the example. No change need be made in the model.

On the other hand, if a positive example is given and this is accepted by the model, the model may or may not need to be changed. Each of the components must be examined *individually*. The class with maximum P_i/N_i^n may or may not be the class with maximum P_i/N_i^{n+1}. The modification of the model is relatively simple because the components are optimized independently of one another.

A similar situation arises if we are given a positive example that is rejected by the model. Each of the component classes in the model that rejects the corresponding example component must be expanded to include the example component. There will usually be many ways to expand and in each case we chose the class for which P_i/N_i^{n+1} is maximum, $n+1$ being the number of positive examples thus far.

Consider next a negative case that is accepted by the model. Each component class in the model can be contracted in various ways so that it will not include the corresponding component of the negative example. For each component, i there will be a satisfactory rejecting class with a maximum P_i/N_i^n.

Let α_i be P_i/N_i^n for the ith component of the model before the negative case occurred. Let α_i' be P_i'/N_i^n for the best acceptable class that can reject the ith component of the new negative case. Note that $\alpha_i' \leq \alpha_i$ for all i. Then we will modify the single component class for which α_i'/α_i is maximum.

The reason is that the likelihood of the model before the negative case was $\prod_i \alpha_i$. We want to replace *only one* of the α_i with its corresponding α_i' and to do this we will choose i such that the new product is maximum — i.e. decreased least. Modifying more than one component would *have* to give a smaller product.

The treatment of these four cases appears to be simpler and more precise than Winston's heuristic discussions on the relative likelihoods of several possible models. No tree exploration is necessary.

The P_i values are readily approximated by counting the relative frequencies of various classes. If the samples are small, the accuracy can be improved by considering how the classes were constructed.

The N_i are obtainable by counting the number of different members of each class that have occurred up to now.

There are two other methods of class construction that can sometimes be more appropriate.

One possibility is to assume that the positive examples are not necessarily typical. The problem is then merely to find the model of highest a priori probability that accepts all known positive and rejects all known negative examples. Instead of P_i/N_i^n , the quantity that we want to maximize is P_i. This is mathematically equivalent to having all of the N_i constant — the same for all classes. The discussions of what to do when a positive case fits the model or when a negative case doesn't are similar to those used before, but if a positive case fits

the model, the model is always left invariant.

The other alternative is to use the more general model of induction given by stochastic languages. Here, the classes usually do not have sharp edges. The description of a stochastic language (which we will identify with a component class in our model) at once gives the a priori probability of that class as well as the probability of any particular element being chosen. The latter corresponds to the $1/N_i$ that was used earlier.

If P_i is as before and a_{ij} is the probability that the ith class assigns to the jth positive example, then we want a class that assigns zero probability to all negative examples such that $P_i \prod_{j=1}^{n} a_{aj}$ is maximum. This criterion corresponds to the earlier approximation P_i/N_i^n

The three methods of class definition described above are not mutually exclusive. It is possible to describe some vector components by means of the P_i/N_i^n model, others by means of the P_i model and still others by means of the stochastic language model.

In the foregoing analysis, we have assumed that the vector components are statistically independent, and the conclusions obtained follow rigorously from this. If there is reason to believe that several components are statistically dependent, a suitable joint a priori probability distribution for them can be obtained. This dependent set would be treated as a single vector component, but the rest of the analysis would be the same.

If we have training sequences with only positive examples, neither Winston's system nor the P_i system can operate, but both the P_i/N_i^n system and the stochastic language system have no particular difficulty.

Another important advantage of the analysis techniques described is the use of quantitative probabilities for the vector classes of the model. From them it is possible to calculate readily the probability that any unidentified new example is or is not an example of the concept being learned.

To compute this, take the total probabilities of all models of the concept that are consistent with the data thus far (i.e. they accept all positive and reject all negative examples) that accept the new example. Divide this by the total probabilities of all models of the concept that are consistent with the data thus far, whether or not they accept the new example.

Quantitative probability estimates are necessary to make critical decisions through decision theory. The mechanization of medical diagnosis is one important area. Decisions in economics, ecology and agriculture are but a few other areas where probabilities of this kind are of paramount importance.

References

[1] Kolmogorov, A.N. "Logical Basis for Information Theory and Probability Theory", *IEEE Transactions on Information Theory* IT–14, pp. 662–604, 1968.

"Three Approaches to the Quantitative Definition of Information", *Information Transmission*, Vol I, pp. 3–11, 1965.

[2] Martin Löf, P. "The Definition of Random Sequences", *Information and Control*, Vol. 9, no. 6, Dec. 1966, pp. 602–619.

[3] Loveland, D.W. "A Variant of the Kolmogorov Concept of Complexity", *Information and Control 15* (1969), pp. 510–526.

[4] Schnorr, C.P. "Process Complexity and Effective Random Tests", *Journal of Computer System Sciences 7* (1973), pp. 376–388.

[5] Zvonkin, A.K., and Levin, L.A. "The Complexity of Finite Objects and the Development of the Concepts of Information and Randomness by Means of the Theory of Algorithms", *Russian Mathematical Surveys 25*, 6 (Nov.–Dec. 1970), pp. 83–124.

[6] Chaitin, G.J. "A Theory of Program Size Formally Identical to Information Theory", To appear in *Journal of the Association of Computing Machinery*.

[7] Cover, T.M. "Universal Gambling Schemes and the Complexity Measures of Kolmogorov and Chaitin", Report No. 12, Statistics Dept., Stanford Univ., Stanford, Calif., 1974. Submitted to *Annals of Statistics*.

[8] Solomonoff, R.J. "A Formal Theory of Inductive Inference." *Information and Control*, March 1964, pp. 1–22 June 1964, pp. 224–254.

[9] Willis., D.G. "Computational Complexity and Probability Constructions", *Journal of the Assoc. of Computing Machinery*, April 1970, pp. 241–259.

[10] Akaike. H. "Information Theory and an Extension of the Maximum Likelihood Principle", *Proc. Second Int. Symp. on Information Theory*, Supplement to *Problems of Control and Information Theory* (1972) pp. 267–281.

[11] Fu, K.S. "Syntactic Methods in Pattern Recognition", Acedemic Press, 1974.

[12] Winston, P.H. "Learning Structural Descriptions from Examples", Report AI-TR231, Artificial Intelligence Lab., Mass. Inst. Technol., 1972. Also can be found in "The Psychology of Computer Vision", edited by P.H. Winston; McGraw Hill, 1974.

[13] Chaitin, G.J. "On the Length of Programs for Computing Finite Binary Sequences", *Journal of the Assoc. of Computing Machinery*, Vol. 13. No. 4 (Oct 1966), pp. 547–569.