Inductive Reasoning and Kolmogorov Complexity

*Ming Li**

University of Waterloo, Department of Computer Science
Waterloo, Ontario N2L 3G1, Canada

Paul M.B. Vitányi

Centrum voor Wiskunde en Informatica, Kruislaan 413,
1098 SJ Amsterdam, The Netherlands
and
Universiteit van Amsterdam, Faculteit Wiskunde en Informatica

ABSTRACT

This is a sloppy first draft of [J. Comp. System Sciences, 44:2(1992), 343-384]. Also, there are some problems with the pictures and references due to the obsolete troff processing.

Reasoning to obtain the 'truth' about reality, from external data, is an important, controversial, and complicated issue in man's effort to understand nature. (Yet, today, we try to make machines do this.) There have been old useful principles, new exciting models, and intricate theories scattered in vastly different areas including philosophy of science, statistics, computer science, and psychology. We focus on inductive reasoning in correspondence with ideas of R.J. Solomonoff. While his proposals result in perfect procedures, they involve the noncomputable notion of Kolmogorov complexity. In this paper we develop the thesis that Solomonoff's method is fundamental in the sense that many other induction principles can be viewed as particular ways to obtain computable approximations to it. We demonstrate this explicitly in the cases of Gold's paradigm for inductive inference, Rissanen's minimum description length (MDL) principle, Fisher's maximum likelihood principle, and Jaynes' maximum entropy principle. We present several new theorems and derivations to this effect. We also delimit what can be learned and what cannot be learned in terms of Kolmogorov complexity, and we describe an experiment in machine learning of handwritten characters. We also give an application of Kolmogorov complexity in Valiant style learning, where we want to learn a concept probably approximally correct in feasible time and examples.

The eye of the understanding is like the eye of the sense; for as you may see great objects through small crannies or levels, so you may see great axioms of nature through small and contemptible instances. [Francis Bacon, *Sylva Sylvarum* 337, 1627]

1. A Historical View of Inductive Reasoning

The Oxford English Dictionary gives as the meaning of **induction:** *the process of inferring a general law or principle from the observations of particular instances*. This defines precisely what we would like to call *inductive inference*. On the other hand, we regard *inductive reasoning* as a more general concept than inductive inference, namely as a process of re-assigning a probability (or credibility) to a law or proposition from the observation of particular instances. In other words, in the way we use the notions, inductive inference draws conclusions that consist in *accepting or rejecting* a proposition, while inductive reasoning only changes the degree of our belief in a proposition. The former is a special case of the latter. In this paper we discuss inductive reasoning in correspondence with R.J. Solomonoff's ideas as expressed in e.g. [Solomonoff 1964 However, Solomonoff's procedure is not effective, since it involves the noncomputable Kolmogorov complexity of objects. We shall show, however, that there is considerable structure in many different approaches proposed for induction, since they can be variously derived as computable approximations to Solomonoff's method.

The history of inductive inference, which is as old as empirical science itself, dates back at least to the Greek philosopher of science Epicurus (342? - 270? B.C). To reason by induction is nothing but to learn from experience. As the sun rises day by day, our belief in that the sun will

* The work of the first author was supported in part by National Science Foundation Grant DCR-8606366, Office of Naval Research Grant N00014-85-k-0445, Army Research Office Grant DAAL03-86-K-0171, and NSERC Operating Grant OGP0036747. Part of the work was performed while he was at the Department of Computer Science, York University, North York, Ontario , Canada. A preliminary form of this paper appeared in *Proc. 4th IEEE Structure in Complexity Theory Conference*, 1989.

rise tomorrow increases, and we eventually infer the *truth* that the sun *will* rise every morning. As human history evolves, man tries to understand and explain the events that happen around him: this takes the form of different induction methods to formulate scientific theories from positive and negative, fortunate and unfortunate, lucky and unlucky, happy and miserable experiences. Two metaphysical principles stand out and prevail today: the principle of Epicurus' multiple explanations (or indifference) and Occam's principle of simplest explanation (Occam's razor).

The **Principle of Multiple Explanations:** *If more than one theory is consistent with the data, keep them all.*

The source of the following material is [Epicurus Epicurus, in his *Letter to Pythocles*, explains that: There are cases, especially of events in the heavens such as the risings and settings of heavenly bodies and eclipses, where it is sufficient for our happiness that several explanations be discovered. In these cases, the events 'have multiple causes of coming into being and a multiple predication of what exists, in agreement with the perceptions.' Epicurus maintains that, if several explanations are in agreement with the (heavenly) phenomena, then we must keep all of them for two reasons. Firstly, the degree of precision achieved by multiple explanations is sufficient for human happiness. Secondly, it would be unscientific to prefer one explanation to another when both are equally in agreement with the phenomena. This, he claims, would be to 'abandon physical inquiry and resort to myth.' His follower Lucretius (95 - 55 B.C.) illustrates the inevitability of the use of the multiple explanation principle by the following example:

> ''There are also some things for which it is not enough to state a single cause, but several, of which one, however, is the case. Just as if you were to see the lifeless corpse of a man lying far away, it would be fitting to state all the causes of death in order that the single cause of this death may be stated. For you would not be able to establish conclusively that he died by the sword or of cold or of illness or perhaps by poison, but we know that there is something of this kind that happened to him.''

Based on the same intuition, in the calculus of probabilities it has been customary to postulate the 'principle of indifference' or the 'principle of insufficient reason'. When there is no other evidence, because of the absolute lack of knowledge concerning the conditions under which a die falls, we have no reason to assume that a certain face has higher probability of turning up. Hence we assume that each side of the die has the probability 1/6. The **principle of indifference** considers events to be equally probable if we have not the slightest knowledge of the conditions under which each of them is going to occur. For the case of a die, this actually coincides with the so-called 'maximum entropy principle', we will discuss later, which states that we should choose probabilities p_i for face i to be the outcome of a trial, $i = 1, 2, ..., 6$, such that $-\sum_{i=1}^{6} p_i \ln p_i$ is maximized under the only constraint $\sum_{i=1}^{6} p_i = 1$. We obtain precisely $p_i = 1/6$ for $i = 1, 2, ..., 6$.

The second and more sophisticated principle is the celebrated Occam's razor principle commonly attributed to William of Ockham (1290? - 1349?). This enters the scene about 1500 years after Epicurus. In sharp contrast to the principle of multiple explanations, it states:

Occam's Razor Principle: *Entities should not be multiplied beyond necessity.*

This is generally interpreted as: Among the several theories that are all consistent with the observed phenomena, one should pick the simplest theory. (According to Bertrand Russell, the actual phrase used by Ockham was: 'It is vain to do with more what can be done with fewer.') Surely Occam's razor principle is easily understood from an 'utilitarian' point of view: if both theories explain the same set of facts, why not use the simpler theory?! However things become more intricate when we want to know whether a simpler theory is really better than the more complicated one. This also raises another question which has been a bone of contention in Philosophy ever since the razor's inception. For what is the proper measure of simplicity? Is $x^{100}+1$ more complicated than $ax^{17}+bx^{2}+cx+d$? E.g., the distinguished contemporary philosopher K. Popper pronounced that the razor is without sense to use on such grounds. However, it is interesting to notice that the principle can be given objective contents, and has recently been very successfully applied in many different forms in computational learning theory.

To explain this, let us consider an over-simplified example of inferring a finite automaton with one-letter input using Occam's razor principle.

Accepted inputs: 0, 000, 00000, 000000000;

Rejected inputs: ε, 00, 000000;

For these data, there exist many consistent finite automata. Figure 1 shows the trivial automaton and Figure 2 shows the smallest automaton, where S denotes starting state and darker states are accepting states.

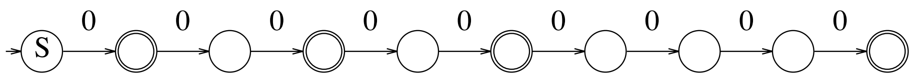

Figure 1: A trivial automaton

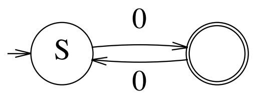

Figure 2: The smallest automaton

Intuitively, the automaton in Figure 1 just encodes data plainly, we therefore do not expect that machine anticipate the future data. On the other hand the second machine makes a plausible *inference* that the language accepted consists of strings of an odd number of 0's. The latter appeals to our intuition as a reasonable inference. However, a too simplistic application of Occam's razor principle may also lead to nonsense as the following story illustrates.

Once upon a time, there was a little girl named Emma. Emma had never eaten a banana, nor had she been on a train. One day she went for a journey from New York to Pittsburgh by train. To relieve Emma's anxiety, her mother gave her a large bag of bananas. At Emma's first bite of a banana, the train plunged into a tunnel. At the second bite, the train broke into daylight again. At the third bite, Lo! into a tunnel; the fourth bite, La! into daylight again. And so on all the way to Pittsburgh and to the bottom of her bag of bananas. Our bright little Emma (applying Occam's razor principle?) told her grandpa: 'Every odd bite of a banana makes you blind; every even bite puts things right again.' [After N.R. Hanson, 'Perception and Discovery', Freeman, Cooper & Co, 1969, p.359.]

Let us consider how the idea of 'simplicity' affects a scientist's thinking. We refer to a beautiful study of simplicity by Kemeny [Kemeny Initially, there were no new facts that failed to be explained by the Special Theory of relativity. The incentive to invent the General Theory of Relativity, by Albert Einstein, was his conviction that the Special Theory was not the *simplest* theory that can explain all the observed facts. Reducing the number of independent variables obviously simplifies a theory. By the requirement of general covariance Einstein succeeded in replacing the previous independent 'gravitational mass' and 'inertial mass' by a single concept.

In spite of the apparent universal acceptance of Occam's razor, consciously or unconsciously, the concept of simplicity remains highly controversial. Generally speaking, it has remained a crude non-precise idea. Things are subtler than they appear. Is the following formulation precise?

Occam's Razor Rule: *Select a hypothesis which is as well in agreement with the observed value as possible; if there is any choice left, choose the simplest possible hypothesis.*

Example. Consider the problem of fitting n points by a polynomial. The above rule tells us to choose the polynomial of lowest degree passing through all the n points. But due to measurement precision and possible noise, whatever degree polynomial the points originated from, we will end up a polynomial of degree $n-1$ which fits the data precisely. But this polynomial most likely does not help us to predict future points.

Example. Consider another example given by Kemeny: Let there be unknown number of white balls and black balls in a sealed urn. Through an opening you randomly pick one ball at a time, note its color and replace it, and shake the urn thoroughly. After n draws you must decide what fracton of the balls in the urn is white. The possible hypotheses state that some rational fraction r of balls in the urn is white, where $0 \leq r \leq 1$. By the above rule, if in n draws, m white balls are selected, then we should formulate the hypothesis $r = m/n$. Let there be 1/3 white and 2/3 black balls. However the probability of getting the true hypothesis $m = n/3$ is zero if n is not divisible by 3, and it tends to zero, even under the assumption that n is divisible by 3. On the other hand we know that to obtain a hypothesis $1/3 - \varepsilon \leq r \leq 1/3 + \varepsilon$, for any ε, has probability tending to 1 exponentially fast, by the so-called Chernoff formula. (For Chernoff's formula see e.g. [Angluin Valiant) Even when the process converges, n may be too large for practical use.

Kemeny's Rule. *Select the simplest hypothesis compatible with the observed values.*

Here 'compatible' is defined as follows. The hypothesis H_i is *compatible* with data D if, assuming the truth of H_i, there was at most one percent chance of getting a deviation as great as $m(H_i, D)$ for some measure function m. This is related to Valiant's learning theory to be discussed later.

But how does one define simplicity? Is 1/4 simpler than 1/10? Is 1/3 simpler than 2/3? Saying that an urn contains 1/3rd part white balls comes down to the same thing as saying that it contains a 2/3rd part black balls. Kemeny warned: 'do not use more precision in your theories than is necessary.' But what is necessary and what is not? All these issues are very subjective. Does a simple theory generate a hypothesis which is good for predicting future outcomes? How do we achieve fast convergence? How does one trade between 'simplicity' and 'truth' ('compatibility')? Kemeny actually asked for 'a criterion combining an optimum of simplicity and compatibility' [crediting Nelson Goodman for this suggestion].

1.1. Combining Epicurus, Ockham, and Bayes

The study of inductive reasoning predates artificial intelligence or computer science by more than 2000 years. There is tremendous amount of literature in many different fields under diverging terminologies. Our goal is to extract a common core of simple ideas underlying all these approaches, in the spirit of Occam's Razor principle. We will start with Bayesian inference theory.

To apply Bayesian type reasoning one has to assign *a priori* probabilities (prior probability) to each possible hypothesis. Since the posthumous publication in 1763 of Th. Bayes' (?? - 1761) famous memoir 'An essay towards solving a problem in the doctrine of chances' by his friend Richard Price, [Bayes essay there has been continuous bitter debate on the controversial prior probability in the Bayesian formula.

The invention of Kolmogorov complexity, by its first inventor R. Solomonoff, was as an auxiliary notion to resolve this particular problem. Namely, using Kolmogorov complexity he found a single 'universal' prior distribution which can be substituted for any particular actually valid distribution (as long as it is computable) in Bayes' formula, and obtain approximately as good results as if the actually valid distribution had been used. It sounds like magic, but Solomonoff's approach does give a more or less satisfactory solution to this unlikely objective.

The elegant idea of a universal prior is a combination of Occam's razor and modern computability theory. However, the universal prior is uncomputable, since it involves Kolmogorov complexity. In this paper we develop the thesis that many theories, models, and principles for inductive reasoning that were formulated both before and after Solomonoff's inception, can be rigorously derived as particular computable approximations to it.

We first describe the basics of Bayesian theory and how to apply Kolmogorov complexity to obtain the Universal prior probability distribution. We then derive the Gold paradigm and its principles. We derive a form of Rissanen's Minimum Description Length (MDL) principle. From the MDL principle Rissanen derives the Fisher's Maximum Likelihood principle and Jaynes Maximum Entropy principle. This paper contains a review of all these theories and principles. It has been our experience that some experts say the connections as claimed are obvious, while some other experts deny those connections exist. Thus, since the proof of the pudding is in the eating, we explicitly establish derivations together with the appropriate related theorems. We also describe an experiment we have performed in machine learning of recognition of handwritten characters using the MDL principle. Combination of Gold-style inference with ideas from computational complexity theory leads to Valiant's model of deductive learning. We give an application of the Universal prior distribution to obtain a theory of learning simple concepts under simple distributions. A more extensive treatment of this material will be given in our forth-coming textbook [Li Introduction Kolmogorov

2. The Universal Prior Distribution

2.1. Bayesian Inference

In the following discussion of probability we assume the usual so-called Kolmogorov Axioms, see e.g. [Feller Introduction For our purpose we need the following. We have a *hypothesis space*, $\mathbf{H} = \{H_1, H_2, \ldots\}$, which consists of a countable set of hypotheses, which are mutually exclusive (in the sense that at most one of them is right), and exhaustive (in the sense that at least one of them is right). With each hypothesis H_i we associate a *probability* $P(H_i)$ such that $\sum_i P(H_i) = 1$. The student is supplied with some data D, providing information about which hypothesis is correct. From the definition of conditional probability, i.e., $P(A \mid B) = P(A \cap B)/P(B)$, it is easy to derive **Bayes' formula** (rewrite $P(A \cap B)$ in the two possible different ways, equate the two expressions, and set $A = H_i$ and $B = D$):

$$P(H_i \mid D) = \frac{P(D \mid H_i) P(H_i)}{P(D)} \tag{1}$$

where

$$P(D) = \sum_i P(D \mid H_i) P(H_i).$$

We interpret the different variables in the formula as follows. The H_i's represent the possible alternative hypotheses concerning the phenomenon we wish to discover. The term D represents the empirically or otherwise known data concerning this phenomenon. The term $P(D)$, the probability of data D, can be considered as a normalizing factor so that $\sum_i P(H_i \mid D) = 1$. The term $P(H_i)$ is called the *prior* probability or *initial* probability of hypothesis H_i, i.e., the probability that H_i is true before we have seen any evidence. The term $P(H_i \mid D)$ is called the *final*, *a posteriori*, or *inferred* probability, which reflects the probability of H_i modified from the prior probability $P(H_i)$ after seeing the data D. The term $P(D \mid H_i)$, the conditional probability of seeing D when H_i is true, is assumed to be computable from D and H_i. In many learning situations, data can only be consistent with an hypothesis H_i in the sense of being forced by it such that $P(D \mid H_i) = 1$. If the data is inconsistent with hypothesis H_i then $P(D \mid H_i) = 0$. In such a situation, the data either is determined by a hypothesis, or disqualifies it. (We assume there is no noise that distorts the data.) For example, the hypothesis is datum $x \in L$ or x L.

The most interesting term is the prior probability $P(H_i)$. In the context of machine learning, $P(H_i)$ is often considered as the learner's *initial degree of belief* in hypothesis H_i. In essence Bayes' rule is a *mapping* from *a priori* probability $P(H_i)$ to *a posteriori* probability $P(H_i \mid D)$, where the mapping is determined by data D. In general, the problem is not so much that in the limit the inferred probability would not 'condense' on the 'true' hypothesis, but that the inferred probability gives as much information as possible about the possible hypotheses from only a limited number of data, cf. example below. In fact, the continuous bitter debate between the Bayesian and non-Bayesian opinions centered on the *prior* probability. The controversy is caused by the fact that Bayesian theory does not say how to initially derive the prior probabilities for the hypotheses. Rather, Bayes' rule only says how they are to be *updated*. However, in each actual case the prior probabilities may be unknown, uncomputable, or conceivably do not exist. (What is the prior probability of use of words in written English? There are many different sources of different social backgrounds living in different ages.) This problem is solved if we can find a *single* probability distribution to use as the prior distribution in each different case, with approximately the same result as if we had used the real distribution. Surprisingly, this turns out to be possible up to some mild restrictions.

Example. We use an example of von Mises [von mises probability truth Let an urn contain many dice each with different attributes. A die with attribute p has probability p showing 6 in a random throw. For convenience, assume the attribute set A is finite, and the difference between each pair of attributes is greater than 2ϵ. Randomly draw a die from the urn, our task is to determine its attribute. We do this by experimenting. Throw the die n times independently. If 6 shows up m times, we choose the attribute that is nearest to m/n. Let H_p be the event of drawing a die with attribute p from an urn. Let D_q be the experimental data such that m successes (6's)

* Properly speaking, formula (1) is not due to Bayes, but it is due to P.S. Laplace (1749 - 1827) who stated the formula and attached Bayes' name to it [Laplace Actually, Bayes in his original paper [Bayes Essay assumed the uniform distribution for the a priori probability, hence he has essentially derived $P(H_i \mid D) = P(D \mid H_i) / \sum_i P(D \mid H_i)$. Although this formula can be derived from (1) by simply assuming that all $P(H_i)$ are the same, Bayes did not state his result in the general form as in (1), nor did he derived his result through a formula similar to (1). Despite the fact that Bayes' rule is just a rewriting of the definition of conditional probability and nothing more, it is its interpretation and applications that are most profound and caused much bitter controversy during the past two centuries.

were observed out of n throws and $|q - m/n| < \varepsilon$, for $q \in A$. So

$$P(H_p | D_q) = \frac{P(D_q | H_p) P(H_p)}{P(D_q)},$$

where $P(D_q) = \sum_p P(D_q \mid H_p) P(H_p)$. According to Chernoff's formula (see [angluin valiant), for $\alpha < 1$, we have:

$$P(m - np > \alpha np \mid H_p) < e^{-\frac{1}{2}\alpha^2 np},$$

$$P(np - m > \alpha np \mid H_p) < e^{-\frac{1}{3}\alpha^2 np}.$$

Hence, if p is the true attribute of the die we have drawn then, choosing $\alpha = p/2\varepsilon$ (so $|p - m/n| \geq \varepsilon$ implies $|m - np| > \alpha np$), $P(D_q \mid H_p)$ goes to 0 exponentially fast when the number of experiments increases, for $q \neq p$, and $P(D_p \mid H_p)$ goes to 1 at the same rate. Hence $P(H_p | D_p)$ goes to 1. Thus we derive the correct attribute of the die with high probability (using a polynomial number of throws). The interesting point is that if the number of trials is small, then the inferred probability $P(H_p | D_q)$ may heavily depend on the the prior probability $P(H_p)$. However, if n is large, then irrespective of the prior distribution, the inferred probability condenses more and more around m/n.

Example. We explain a simplified version of Solomonoff's theory of inductive inference. The simplification is in that we, for the moment, consider only a discrete sample space like $\{0, 1\}^*$, The set of all finite binary sequences, rather than $\{0, 1\}^\infty$, the set of all one-way infinite binary sequences.

We view theory formation in science as the process of obtaining a compact description of the past observations together with predictions of future ones. The investigator observes increasingly larger initial segments of an finite binary sequence as the outcome of an finite sequence of experiments on some aspect X of nature. To describe the underlying regularity of this sequence, the investigator tries to formulate a theory that governs X, on the basis of the outcome of past experiments. Candidate theories (hypotheses) are identified with computer programs that compute binary sequences starting with the observed initial segment.

First assume the existence of a prior probability distribution, described by probability function P, over a discrete sample space $\Omega = \{0,1\}^*$. Define a function μ over Ω by $\mu(x) = \sum \{P(xy) : y \in \Omega\}$. Thus, $\mu(x)$ is the probability of a sequence starting with x. Given a previously observed data string S, the inference problem is to predict the next symbol in the output sequence, i.e., extrapolating the sequence S. In terms of the variables in Formula (1), H_i is the hypothesis that the sequence under consideration starts with initial segment $S\,a$. The data D consist in the assertion that the sequence in fact starts with initial segment S. Thus, for $P(H_i)$ and $P(D)$ in Formula (1) we substitute $\mu(S\,a)$ and $\mu(S)$, respectively, and obtain, $a = 0$ or $a = 1$,

$$P(Sa \mid S) = \frac{P(S \mid Sa)\,\mu(Sa)}{\mu(S)}.$$

Here $\mu(S) = P(S \mid S0)\,\mu(S0) + P(S \mid S1)\,\mu(S1) + P(S)$. Obviously, $P(S \mid Sa) = 1$ for any a, hence

$$P(Sa \mid S) = \frac{\mu(Sa)}{\mu(S)}. \tag{2}$$

In terms of inductive inference or machine learning, the final probability $P(Sa \mid S)$ is the probability of the next symbol being a, given the initial sequence S. Obviously we now only need the prior probability to evaluate $P(Sa \mid S)$.

The goal of inductive inference in general is to be able to either (i) predict (extrapolate) the next element of S, or (ii) to infer an underlying effective process (in the most general case, a Turing machine, according to the Church-Turing thesis) that generated S, and hence to be able to predict the next symbol.

In order to solve the problem for unknown prior probability, Solomonoff proposed what he called a **universal prior distribution**. We now carefully define the universal prior distribution and prove several fundamental theorems due to Solomonoff and L.A. Levin, and afterwards continue this example. The definitions and theorems are so fundamental that our approach totally rests upon them. These results are in some form hidden in [Solomonoff 1964] [Solomonoff IEEE 1978 convergence] [Levin Zvonkin] [Gács Lecture Notes 1987 For various reasons they are difficult to access, and almost unknown except to a few people doing research in this area. It seems useful to recapitulate them. First we need the basic definitions of Kolmogorov complexity.

2.2. Kolmogorov Complexity

Inductive reasoning was the midwife that stood at the cradle of Kolmogorov complexity. Nowadays, Kolmogorov complexity has been applied in many areas of computer science and mathematics (see [Kolmogorov complexity handbook for a general survey), and few realize that Kolmogorov complexity was at first invented for the purpose of inductive inference. In this essay, we go back to this origin.

We are interested in defining the complexity of a concrete individual finite string of zeros and ones. Unless otherwise specified, all strings will be binary and of finite length. All logarithms in this paper are base 2, unless it is explicitly noted they are not. If x is a string, then $l(x)$ denotes the *length* (number of zeros and ones) of x. We identify throughout the xth finite binary string with the natural number x, according to the correspondence:

$$(\varepsilon, 0), (0, 1), (1, 2), (00, 3), (01, 4), (10, 5), \ldots$$

Intuitively, we want to call a string simple if it can be described in a few words, like "the string of a million ones"; A string is considered complex if it cannot be so easily described, like a "random" string which does not follow any rule and hence we do not know how to describe apart from giving it literally. A description of a string may depend on two things, the decoding method (the machine which interprets the description) and outside information available (input to the machine). We are interested in descriptions which are effective, and restrict the decoders to Turing machines. Without loss of generality, our Turing machines use binary input strings which we call *programs*. More formally, fixing a Turing machine T, we would like to say that p is a description of x if, on input p, T outputs x. It is also convenient to allow T to have some extra information y to help to generate x. We write $T(p,y)=x$ to mean that Turing machine T with input p and y terminates with output x.

Definition 1. The descriptional complexity C_T of x, *relative* to Turing machine T and binary string y, is defined by

$$C_T(x \mid y) = \min\{l(p): p \in \{0,1\}^*,\ T(p,y)=x\},$$

or ∞ if no such p exists.

The complexity measure defined above is useful and makes sense only if the complexity of a string does not depend on the choice of T. Therefore the following simple theorem is vital. This Invariance Theorem is given by Solomonoff [Solomonoff formal inductive inference Kolmogorov [Kolmogorov Three approaches and Chaitin [Chaitin Ch2

Theorem 1. *There exists a universal Turing machine U, such that, for any other Turing machine T, there is a constant c_T such that for all strings x, y, $C_U(x \mid y) \le C_T(x \mid y) + c_T$.*

Proof. Fix some standard enumeration of Turing machines $T_1, T_2, \cdots$. Let U be the Universal Turing machine such that when starting on input $0^n 1p$, $p \in \{0,1\}^*$, U simulates the nth

Turing machine T_n on input p. For convenience in the proof, we choose U such that if T_n halts, then U first erases everything apart from the halting contents of T_n's tape, and also halts. By construction, for each $p \in \{0,1\}^*$, T_n started on p eventually halts iff U started on $0^n 1p$ eventually halts. Choosing $c_T = n+1$ for T_n finishes the proof. □

Clearly, the Universal Turing machine U that satisfies the Invariance Theorem is *optimal* in the sense that C_U minorizes each C_T up to a fixed additive constant (depending on U and T). Moreover, for each pair of Universal Turing machines U and U', satisfying the Invariance Theorem, the complexities coincide up to an additive constant (depending only on U and U'), for all strings x, y:

$$|C_U(x \mid y) - C_{U'}(x \mid y)| \leq c_{U,\,U'}\,.$$

Therefore, we set the canonical *conditional Kolmogorov* complexity $C(x \mid y)$ of x under condition of y equal to $C_U(x \mid y)$, for some fixed optimal U. We call U the *reference* Turing machine. Hence the Kolmogorov complexity of a string does not depend on the choice of encoding method and is well-defined. Define the *unconditional Kolmogorov* complexity of x as $C(x) = C(x \mid \varepsilon)$, where ε denotes the empty string ($l(\varepsilon) = 0$).

Definition 2. In the sequel we need to use the *prefix* complexity variant, or self-delimiting complexity, rather than $C(x)$ from Definition 1. A *prefix machine* is a Turing machine with three tapes: a one-way input tape, a one-way output tape, and a two-way work tape. Initially, the input tape contains an indefinitely long sequence of bits. If the machine halts, then the initial segment on the input tape it has read up till that time is considered the *input* or *program*, and the contents of the output tape is the *output*. Clearly, the set of programs of each such machine is a prefix-code. (Recall that if p and q are two code words of a *prefix-code*, then p is not a proper prefix of q.) We can give an effective enumeration of all prefix machines in the standard way. Then the *prefix* descriptional complexity of $x \in \{0,1\}^*$, with respect to prefix machine T, and binary string y, is defined as

$$K_T(x \mid y) = \min\{l(p) : p \in \{0,1\}^*,\ T(p,y) = x\},$$

or ∞ if such p do not exist. One can prove an Invariance Theorem for prefix complexity, and define the conditional and unconditional *prefix Kolmogorov complexity*, by fixing some reference optimal prefix machine, in exactly the same way as before, so we do not repeat the construction.

Remark. The prefix Kolmogorov complexity of string x, is the length of the shortest prefix program that outputs x. In this exposition, we will use $K(x)$ to denote the prefix Kolmogorov complexity of x. $C(x)$ and $K(x)$ differ by at most a $2 \log K(x)$ additive term. In some applications this does not make any difference. But in some other applications, for example inductive inference, this is vital. In particular, we need the property that the series $\sum_x 2^{-K(x)}$ converges, cf. below.

Definition 3. A binary string x is *incompressible* if $K(x) \geq l(x)$.

Remark. Since Martin-Löf [Martin-Löf definition random 1966 has shown that incompressible strings pass all effective statistical tests for randomness, we will also call incompressible strings *random* strings. A simple counting argument shows that most strings are random. The theory of computability shows that the function $K(x)$ is noncomputable, but can be approximated to any degree of accuracy by a computable function. However, at no point in this approximation process can we know the error. Cf. also the surveys [Levin Zvonkin] [Li Two Decades

2.3. Semicomputable Functions and Measures

We consider recursive functions with values consisting of pairs of natural numbers. If $<p, q>$ is such a value then we interpret this value as the rational number p/q, and say that the recursive function is *rational valued*.

Definition. A real function f is *semicomputable from below* iff there exists a recursive function $g(x, k)$ with rational values (or, equivalently, a computable real function $g(x, k)$), nondecreasing in k, with $f(x) = \lim_{k \rightarrow \infty} g(x, k)$. A function f is semicomputable from above, if $-f$ is semicomputable from below.

(An equivalent definition: f is a function that is semicomputable from below if the set $\{(x, r): r \leq f(x), r \text{ is rational}\}$ is recursively enumerable.)

A real function f is computable iff there is a recursive function $g(x, k)$ with rational values, and $|f(x) - g(x, k)| < 1/k$.

Obviously, all recursive functions are computable, and all computable functions are semicomputable. However, not all semicomputable functions are computable, and not all computable functions are recursive. Nontrivial examples of functions that are semicomputable from above but not computable are $C(x)$, $C(x \mid y)$, $K(x)$, and $K(x \mid y)$.

The following analysis is a simplified version over the discrete space N (or the set of finite binary strings), of Zvonkin and Levin [Levin Zvonkin We follow to some extent [Gács Lecture

Notes Functions $\mu : N \rightarrow [0, 1]$ that satisfy the usual properties of probability distributions except that

$$\sum_x \mu(x) \leq 1.$$

we shall call *measures*. We say that a measure μ (multiplicatively) *dominates* a measure μ' if there exists a constant c such that, for all x in N, we have $\mu'(x) \leq c\, \mu(x)$. It is known from the calculus that *no* measure μ dominates all measures: for each measure μ there is a measure μ' such that $\lim \mu'(x)/\mu(x) = \infty$. However, if we restrict ourselves to the class of semicomputable measures, then it turns out that this class contains an 'absorbing' element, a measure that dominates all measures in the class. We call the measure that dominates all other measures in a given class a *universal measure* for that class. This important observation that such a measure exists was first made by Levin [Levin Zvonkin

Theorem 2. *The class of measures that are semicomputable from below contains a universal measure.*

Proof. First we consider the standard enumeration of all partial recursive functions $\phi_1, \phi_2, \ldots$. Each $\phi = \phi_i$ in this list is a function on the positive integers. Let $<\cdot>$ denote a standard effective invertible pairing function over N to associate a unique natural number $<x, k>$ with each pair (x, k) of natural numbers. This way we can interpret ϕ as a two-argument function $\phi(<x, k>)$. We change each ϕ into a partial recursive function ψ with the same range as ϕ but with, for each x, the value of $\psi(<x, k>)$ is defined only if $\psi(<x, 1>), \psi(<x, 2>), \ldots, \psi(<x, k-1>)$ are defined. (Assign values to arguments in enumeration order.) We use each ψ to define a semicomputable real function s by rational valued approximations $s^k(x)$, $k = 1, 2, \ldots$, from below:

$$s(x) = \sup\{s^k(x) : s^k(x) = p/q,$$

$$\psi(<x, k>) = <p, q>, k = 1, 2, \ldots\}.$$

The resulting s-enumeration contains all semicomputable functions. Next we use each semicomputable function s to compute a measure μ from below. Initially, set $\mu(x) = 0$ for all x. If $s(1)$ is undefined then μ will not change any more and it is trivially a measure. Otherwise, for $k = 1, 2, \ldots$, if $s^k(1) + s^k(2) + \ldots + s^k(k) \leq 1$ then set $\mu(i) := s^k(i)$ for $i = 1, 2, \ldots, k$, else the computation of μ is finished.

There are three mutually exclusive ways the computation of μ can go, exhausting all possibilities. Firstly, s is already a measure and $\mu := s$. Secondly, for some x and k with $x \leq k$ the value

$s^k(x)$ is undefined. Then the values of μ do not change any more from $\mu(i) = s^{k-1}(i)$ for $i = 1, 2, \ldots, k-1$, and $\mu(i) = 0$ for $i \geq k$, even though the computation of μ goes on forever. Thirdly, there is a first k such that $s^k(1) + s^k(2) + \ldots + s^k(k) > 1$, that is, the new approximation of μ violates the condition of measure. Then the approximation of μ is finished as in the second case. But in this case the algorithm terminates, and μ is even computable.

Thus, the above procedure yields an effective enumeration $\mu_1, \mu_2, \ldots$ of all semicomputable measures. Define the function μ_0 as:

$$\mu_0(x) = \sum_n 2^{-n} \mu_n(x).$$

It follows that μ_0 is a measure since

$$\sum_x \mu_0(x) = \sum_n 2^{-n} \sum_x \mu_n(x) \leq \sum_n 2^{-n} = 1.$$

The function μ_0 is also semicomputable from below, since $\mu_n(x)$ is semicomputable from below in n and x. (Use the universal partial recursive function ϕ_0 and the construction above.) Finally, μ_0 dominates each μ_n since $\mu_0(x) > 2^{-n} \mu_n(x)$. Therefore, μ_0 is a universal semicomputable measure. □

Obviously, there are countably infinite universal semicomputable measures. We now fix a *reference* universal semicomputable measure $\mu_0(x)$, and denote it by $\mathbf{m}(x)$. It will turn out that function $\mathbf{m}(x)$ adequately captures Solomonoff's envisioned universal *a priori* probability.

If a semicomputable measure is also a probability distribution then it is computable. Namely, if we compute an approximation μ^k of the function μ from below for which $\sum_x \mu^k(x) > 1 - \varepsilon$, then we have $|\mu(x) - \mu^k(x)| < \varepsilon$ for all x.

Any positive function $w(x)$ such that $\sum_x w(x) \leq 1$ must converge to zero. Hence $\mathbf{m}(x)$ converges to zero as well. However, it converges to zero slower than any positive computable function that converges to zero. That is, $\mathbf{m}(x)$ is not computable, and therefore it is not a proper probability distribution: $\sum_x \mathbf{m}(x) < 1$. There is no analogous result to Theorem 2 for computable measures: amongst all computable measures there is no universal one. This fact is one of the reasons for introducing the notion of semicomputable measures.

2.4. The Solomonoff-Levin Distribution

The original incentive to develop a theory of algorithmic information content of individual objects was Ray Solomonoff's invention of a *universal a priori* probability that can be used instead of the actual (but unknown) *a priori* probability in applying Bayes' Rule. His original suggestion was to set the *a priori* probability $P(x)$ of a finite binary string x to $\sum 2^{-l(p)}$, the sum taken over all programs p with $U(p)=x$ where U is the reference Turing machine of Theorem 1 for the C-complexity. However, using plain Turing machines this is improper, since not only does $\sum_x P(x)$ diverge, but for some x even $P(x)$ itself diverges. To counteract this defect, Solomonoff in 1960 and 1964 used normalizing terms, but the overall result was unconvincing. Levin [Levin Zvonkin succeeded in 1970 to find a proper mathematical expression of the *a priori* probability, of which we present the simpler version over the discrete domain N. This was elaborated by Levin in 1973 and 1974 [Levin Notion random] [Levin non-growth and Levin and Gács in 1974 [Gács symmetry and independently by Chaitin in 1975 [Chaitin theory 1975 formally identical

Definition. The *Solomonoff-Levin distribution* (actually a measure) on the positive integers is defined by $P_U(x)=\sum 2^{-l(p)}$, where the sum is taken over all programs p for which the reference prefix-machine U of Theorem 1 outputs x. This is a measure because of the following.

Kraft's Inequality. If $l_1, l_2, ...$ is a sequence of positive integers such that $\sum_n 2^{-l_n} \leq 1$ then there is a prefix-code $c: N \rightarrow \{0,1\}^*$ (i.e., if $n \neq m$ are positive integers, then $c(n)$ is not a prefix of $c(m)$), with $l(c(n))=l_n$. Conversely, if $c: N \rightarrow \{0,1\}^*$ is a prefix-code, then the sequence $l_1, l_2, ...$ with $l_n = l(c(n))$, $n=1,2,...$ satisfies the inequality above. See e.g. [Gallager

Hence, by the Kraft Inequality, for the prefix-code formed by the programs p of U we have $\sum_p 2^{-l(p)} \leq 1$. Therefore, the combined probability $\sum_x P_U(x)$, summed over all x's, sums up to less than one, no matter how we choose reference U, because for some program q there is no output at all.

Another way to conceive of $P_U(x)$ is as follows. We think of the input to the reference prefix machine U as being provided by indefinite long sequences of fair coin flips. Thus, the probability of generating a program p for U is $P(p)=2^{-l(p)}$ where P is the standard 'coin-flip' uniform measure. (Namely, presented with any infinitely long sequence starting with p, the machine U, being a prefix-machine, will read exactly p and no further.) Due to the halting problem, for some q the reference U does not halt. Therefore, the *halting probability* Ω satisfies

$$\Omega = \sum_x P_U(x) < 1.$$

Now we are ready to state the remarkable and powerful fact that Levin's universal semicomputable measure $\mathbf{m}(x)$, the Solomonoff-Levin universal *a priori* probability $P_U(x)$, and the simpler expression $2^{-K(x)}$, all coincide up to an independent fixed multiplicative constant. It is a consequence of universally accepted views in mathematical logic (Church's Thesis), that the widest possible effective notion of simplicity of description of an object x is quantified by $K(x)$.

The Solomonoff-Levin distribution can be interpreted as an recursively invariant notion that is the formal representation of ‘‘Occam's Razor'': the statement that one object is simpler than the other is equivalent to saying that the former object has higher probability than the latter.

Theorem 3. *There is a constant c such that for each x, up to additive constant c, we have* $-\log \mathbf{m}(x) = -\log P_U(x) = K(x)$.

Proof. Since $2^{-K(x)}$ represents the contribution to $P_U(x)$ by a shortest program for x, $2^{-K(x)} \le P_U(x)$ for all x. Since $P_U(x)$ is semicomputable from below by enumerating all programs for x, we have by the universality of $\mathbf{m}(x)$ that there is a fixed constant c such that for all x we have $P_U(x) \le c\, \mathbf{m}(x)$.

It remains to show that $\mathbf{m}(x) = O(2^{-K(x)})$. This is equivalent to showing that for some constant c we have $-\log \mathbf{m}(x) \ge K(x) + c$. It suffices to exhibit a prefix code such that for some other fixed constant c', for each x there is a code word p such that $l(p) \le -\log \mathbf{m}(x) + c'$, together with a prefix-machine T such that $T(p) = x$. Then, $K_T(x) \le l(p)$ and hence by the Invariance Theorem 1 also $K(x) \le l(p)$ up to a fixed additive constant. First we recall a construction for the Shannon-Fano code.

Claim. If μ is a measure on the integers, $\sum_x \mu(x) \le 1$, then there is a binary prefix-code $r : N \rightarrow \{0, 1\}^*$ such that the code words $r(1), r(2), \ldots$ are in lexicographical order, such that $l(r(x)) \le -\log \mu(x) + 2$. This is the Shannon-Fano code.

Proof. Let $[0, 1)$ be the half open unit real interval. The half open interval $[0.x, 0.x + 2^{-l(x)})$ corresponding to the set (cylinder) of reals $\Gamma_x = \{0.y : y = x\, z\}$ (x finite and y and z infinite binary strings) is called a *binary interval*. We cut off disjoint, consecutive, adjacent (not necessarily binary) intervals I_n of length $\mu(n)$ from the left end of $[0, 1)$, $n = 1, 2, \ldots$. Let i_n be the length of the longest binary interval contained in I_n. Set $r(n)$ equal to the binary word corresponding to the first such interval. It is easy to see that I_n is covered by at most four binary intervals of length i_n, from which the claim follows. □

Since $\mathbf{m}(x)$ is semicomputable from below, there is a partial recursive function $\phi(t, x)$ such that $\phi(t, x) \leq \mathbf{m}(x)$ for all t, and $\lim_{t \to \infty} \phi(t, x) = \mathbf{m}(x)$. Let $\psi(t, x) = 2^{-k}$, with k is a positive integer, be the greatest partial recursive lower bound of this form on $\phi(t, x)$. We can assume that ψ enumerates its range without repetition. Then,

$$\sum_{x,t} \psi(t, x) = \sum_x \sum_t \psi(t, x) \leq \sum_x 2\,\mathbf{m}(x) \leq 2.$$

(The series $\sum_t \psi(t, x)$ can only converge to precisely $2\,\mathbf{m}(x)$ in case there is a positive integer k such that $\mathbf{m}(x) = 2^{-k}$.)

Similar to before, we chop off consecutive, adjacent, disjoint half open intervals $I_{t,x}$ of length $\psi(t, x)/2$, in order of computation of $\psi(t, x)$, starting from the left side of $[0, 1)$. This is possible by the last displayed equation. It is easy to see that we can construct a prefix-machine T as follows. If Γ_p is the largest binary interval of $I_{t,x}$, then $T(p) = x$. Otherwise, $T(p)$ is undefined (e.g., T doesn't halt).

By construction of ψ, for each x there is a $\psi(t, x) > \mathbf{m}(x)/2$. By the construction in the Claim, each interval $I_{t,x}$ has length $\psi(t, x)/2$. Each I-interval contains a binary interval Γ_p of length at least one quarter of that of I. Therefore, there is a p with $T(p) = x$ such that $2^{-l(p)} \geq \mathbf{m}(x)/16$. This implies $K_T(x) \leq -\log \mathbf{m}(x) + 4$. The proof of the theorem is finished. □

Theorem 3 demonstrates a particularly important instance of the two conceptually different, but equivalent, definitions of the semicomputable measures. We analyse this equivalence in some detail. Let $P_1, P_2, \ldots$ be the effective enumeration of all semicomputable probability distributions constructed in Theorem 2. Let $T_1, T_2, \ldots$ be the standard enumeration of prefix-machines. For each prefix-machine T, define

$$Q_T(x) = \sum_{T(p) = x} 2^{-l(p)}.$$

In other words, $Q_T(x)$ is the probability that T computes output x if its input p is generated by successive tosses of a fair coin. In other words, the inputs p are uniformly distributed with the probability of p occurring equal $2^{-l(p)}$. It is easy to see that each Q_T satisfies

$$\sum_x Q_T(x) \leq 1.$$

Equality holds iff T halts for all inputs (proper programs). Let $Q_1, Q_2, \ldots$ (where we do not require equality to hold) be the probability distributions associated with $T_1, T_2, \ldots$.

Claim. There are recursive functions σ, π such that $Q_n = \Theta(P_{\sigma(n)})$ and $P_n = \Theta(Q_{\pi(n)})$, for $n = 1, 2, \ldots$.

Proof. Omitted. □

Remark. The Coding Theorem tells us that there is a constant $c > 0$ such that $-\log P_U(x) - K(x) \leq c$. We recall from the definition of the Solomonoff-Levin distribution that $-\log P_U(x) = -\log \sum_{U(p)=x} 2^{-l(p)}$, and $K(x) = \min\{l(p) : U(p) = x\}$. *A priori* an outcome x may have high probability because it has many long descriptions. But these relations show that in that case it must have a short description too. In other words, the *a priori* probability of x is governed by the shortest program for x.

Remark. Let P be any probability distribution (not necessarily computable). The P-expected value of $\mathbf{m}(x)/P(x)$ is

$$\sum_x P(x) \frac{\mathbf{m}(x)}{P(x)} < 1.$$

We find by Chebychev's first Inequality[1)] that

$$\sum \{P(x) : \mathbf{m}(x) \leq k\, P(x)\} \geq 1 - \frac{1}{k}. \tag{3}$$

Since $\mathbf{m}(x)$ dominates all semicomputable measures multiplicatively, for all x we have

$$P(x) \leq c_P\, \mathbf{m}(x), \tag{4}$$

for a fixed positive constant c_P independent of x (but depending on the index of P in the effective enumeration $\mu_1, \mu_2, \ldots$ of semicomputable measures).

Inequalities (3) and (4) have the following consequences:[2)]

(i) If x is a random sample from a simple computable distribution $P(x)$, then $\mathbf{m}(x)$ is a good estimate of $P(x)$.

(ii) If we know or believe that x is random with respect to P, and we know $P(x)$, then we can use $P(x)$ as an estimate of $\mathbf{m}(x)$.

1) Recall that *Chebychev's First Inequality* says the following. Let P be any probability distribution, f any nonnegative function with expected value $E_P(f) = \sum_x P(x) f(x) < \infty$. For $\lambda \geq 0$ we have $\sum\{P(x) : f(x) > \lambda\} < \lambda^{-1} E_P(f)$. Here we use it with $k\, E_P(f)$ substituted for λ.

2) We shortly remark, without further explanation, that in both cases the degree of approximation depends on the index of P, and the randomness of x with respect to P, as measured by the randomness deficiency $\delta_0(x \mid P) = \log(\mathbf{m}(x)/P(x))$. If $\delta_0(x \mid P) = O(1)$ then x is random, otherwise x is not random. For example, for the Uniform Distribution

2.4.1. Solomonoff's Inference Procedure and Its Mathematical Justification

We continue Solomonoff's approach to inductive inference, as in [Levin Zvonkin In general, one cannot prove that an inference procedure in statistics is good. This accounts for the many different approaches which are advocated in statistics. In contrast, about Solomonoff's procedure we can rigourously prove that it is good. First, we put the previously developed theory in a continuous setting. Let the sample space $S = \{0, 1\}^* \cup \{0, 1\}^\infty$, the set of all finite and one-way infinite binary sequences. Let a cylinder $\Gamma_x = \{xy : y \in S\}$, the set of all elements from S that start with x. A function μ from cylinders to the real interval [0, 1] is called a *semimeasure* if

(a) $\mu(S) \leq 1$; and

(b) $\mu(\Gamma_x) \geq \mu(\Gamma_{x0}) + \mu(\Gamma_{x1})$.

A semimeasure is called a *measure* if equality holds in (a) and (b). A semimeasure μ is *(semi)computable* if $f(x) = \mu(\Gamma_x)$ is (semi)computable. Note that f needs to satisfy (a) and (b). It is more convenient, and customary in this area, to simply write $\mu(x)$ instead of $\mu(\Gamma_x)$. The problem was that the proper a priori probabilities μ in formula (2) are not known.

We modify the Turing machines in the standard enumeration so that they correspond to the semicomputable measures.

A *monotonic* machine M is a three tape machine similar to the prefix machine we defined before, but now for all finite (binary) inputs p and q, if p is a prefix of q, then $M(q) = M(p)r$ for some r in S. (For convenience we define the $M(p)$ as the contents of the output tape when M reads the next symbol after p. If M doesn't halt then $M(p)$ can be finite or one-way infinite.) Let U be the *universal* monotonic machine, in the same way as we have already met universal Turing machines and universal prefix machines.

The *universal semicomputable semimeasure* is defined as

$$\mathbf{M}(x) = \sum_{U(p) \in \Gamma_x} 2^{-l(p)},$$

i.e., $\mathbf{M}(x)$ is the *a priori probability* that the output of the reference monotonic machine U starts with x. Just as in the discrete case, one can show that for each semicomputable semimeasure μ,

$\delta_0(x \mid P) = n - K(x \mid n) + O(1)$, where $n = l(x)$. Such a (universal Martin-Löf) test is needed, since otherwise we cannot distinguish, for instance, between randomness and nonrandomness of samples from the uniform distribution. (Clearly, the word C o n s t a n t i n o p l e is not a random 14-letter word. The probability of seeing it somewhere written is decidedly greater than 128^{-14}, say, for a randomly selected fourteen letter ASCII word.)

there exists a constant c, such that for all $x \in \{0, 1\}$, we have

$$\mathbf{M}(x) \geq c\ \mu(x).$$

An alternative approach to defining *a priori probability* was taken by Cover [Cover gambling schemes who defined

$$\mathbf{Mc}(x) = \sum \{\mathbf{m}(xy) : y \in \{0, 1\}^*\}.$$

This function has related properties to **M**.

Solomonoff's Predictor. Instead of using formula (2), we estimate the conditional probability $P(xy \mid x)$ that the next segment after x is y by the expression

$$\frac{\mathbf{M}(x\,y)}{\mathbf{M}(x)}. \tag{5}$$

Now let μ in Formula (2) be an arbitrary computable measure. This case includes all computable sequences, as well as many Bernouilli sequences.

Justification. Solomonoff [Solomonoff IEEE 1978 showed that convergence of the error made by the estimator is very fast, in the following sense. If μ is the actual prior probability (measure) over the sample space $\{0, 1\}$, than we obviously cannot do better in predicting a '0' or '1' after an initial segment x than using the inferred probability

$$\frac{\mu(xa)}{\mu(x)},\ a = 0, 1.$$

To estimate how much worse it is to use **M** instead of μ we consider the difference in inferred probabilities. Let S_n denotes the μ-expectation of the squared difference between the μ-inferred probability and the **M**-inferred probability, of '0' occurring as the $n+1$th symbol:

$$S_n = \sum_{l(x)=n} \mu(x) \left[\frac{\mathbf{M}(x\,0)}{\mathbf{M}(x)} - \frac{\mu(x\,0)}{\mu(x)} \right]^2 .$$

Then $\sum_n S_n \leq K(\mu)/2$. Here, $K(\mu)$ is the Kolmogorov complexity of the index i, where T_i is a Turing machine computing μ. Therefore, S_n converges to zero faster than $1/n$. In other words, it has been rigorously proved that for the above estimator the *expected error* at the nth prediction converges to zero faster than $1/n$!

This was improved by Gács [%A P. Gács %T Personal Communication as follows. If the length of y is fixed, and the length of x grows to infinity, then

$$\frac{\mathbf{M}(x\,y)/\mathbf{M}(x)}{\mu(x\,y)/\mu(x)} \to 1,$$

with μ-probability one. In other words, the conditional a priori probability is almost always asymptotically equal to the conditional probability.

With respect to the discrete sample space approach taken before, one can show that:

$$-\log \mathbf{M}(x) = -\log \mathbf{Mc}(x) = K(x) + O(\log K(x)). \tag{6}$$

2.4.2. Conclusions

On the positive side we have achieved the following. Bayes' rule using the universal prior distribution gives an objective interpretation to Occam's razor principle. Namely, if several programs could generate $S0$ then the shortest one is used (for the prior probability), and further if $S0$ has a shorter program than $S1$ then $S0$ is preferred (*i.e.* predict 0 with higher probability than predicting 1 after seeing S). Bayes' rule via the universal prior distribution also satisfies Epicurus' multiple explanations dictum, since we do not select a single hypothesis after considering the evidence, but maintain all hypotheses consistent with the evidence and just transform the probability distribution on the hypotheses according to the evidence. Finally, there is mathematical proof that Solomonoff's inference procedure using the universal prior probability performs almost as good as the one using the actual (computable) prior probability.

On the negative side we know that Solomonoff's inference is not practicable in its pure form. The universal prior distributions $\mathbf{m}(x)$ for discrete sample spaces, and $\mathbf{M}(x)$ for continuous sample spaces, are not computable, essentially because the Kolmogorov complexity is not computable. However, we can compute approximations to $K(x)$, $\mathbf{m}(x)$, and $\mathbf{M}(x)$. It turns out that using Solomonoff's inference principles with such computable approximations yields many other known inference models or principles. In the next few sections, we derive or establish connections with various well-known machine learning models and inductive inference paradigms or principles. Thus we provide an alternative view of these models and principles from the lofty perspective of Kolmogorov complexity.

3. Gold's Inductive Inference Paradigm

There are many different ways of formulating concrete inductive inference problems in the real world. We will try to simplify matters as much as possible short of losing significance.

(i) **The class of rules** we consider can be various classes of languages or functions, where we restrict ourselves to classes of recursive sets, context-free languages, regular sets and sets of finite automata, and sets of Boolean formulae. We treat a language L as a *function* f using its characteristic function, *i.e.*, $f(x) = \chi_L(x) = 1$ if $x \in L$, and 0 otherwise.

(ii) **The hypothesis space** or **rule space** denoted by **R** specifies syntactically how each rule in (i) should be represented. We fix a standard enumeration of the representations for a class of rules, $\mathbf{R} = \{R_1, R_2, ...\}$, and assume that each rule f has at least one description in the corresponding hypothesis space. For example, the hypothesis space can be standard encodings of context-free grammars, or standard encodings of Finite Automata. In any case, it is assumed that the hypothesis space is effectively enumerable (so it cannot be the set of all halting Turing machine codes). For convenience, this enumeration of hypotheses $R_1, R_2, ...$ consists of codes for algorithms to compute recursive functions $f_1, f_2, ...$ (languages are represented by their characteristic functions).

(iii) **The presentation of examples** is vital to the inference process. We choose the simplest, and yet most general, form of data presentation. For a function f to be inferred, there is a fixed infinite sequence of *examples* $(s_1, f(s_1)), (s_2, f(s_2)), ...$ When $f = \chi_L$, we have $\chi_L(s) = 1$ if $s \in L$ (s is a positive example of L) and $\chi_L(s) = 0$ otherwise (s is a negative example of L).

A rule (or function) f is said to be *consistent* with the initial segment of examples

$$S = (s_1, a_1), \ldots, (s_n, a_n), \tag{7}$$

if $f(s_i) = a_i$, $i = 1,..,n$. We require that *all* strings will eventually appear as first component in a pair in S. The last assumption is strong, but is essential to the Gold paradigm.

How to infer a rule. By (ii), there is an effective enumeration $f_1, f_2, ...$ of partial recursive functions corresponding to the enumeration of hypotheses. The a priori probability of f_k is $\mathbf{m}(f_k) = \mathbf{m}(k)$. (Actually, $\mathbf{m}(f_k) = c\,\mathbf{m}(k)$, for some constant c depending on the effective enumeration involved, but not depending on n. To assume that $c = 1$ makes no difference in the following discussion.) We are given an infinite sequence of examples representing the rule or function f to be learned. According to Bayes' rule (1), for $k = 1, 2, ...$, the inferred probability of f_k after the sequence of examples (7) is given by:

$$P(f_k = f \mid f(s_i) = a_i, i = 1,..,n) = \frac{P(f(s_i) = a_i, i = 1,..,n \mid f_k = f)\,\mathbf{m}(k)}{\sum\{\mathbf{m}(j): f_j(s_i) = a_i, i = 1,..,n\}}. \tag{8}$$

Cf. also [Cover 1974 gambling] [Cover Impact In the numerator of the right-hand term, the first factor is zero or one depending on whether f_k is consistent with S, and the second factor is the a priori probability of f_k. The denominator is a normalizing term giving the combined a priori probability of all rules consistent with S. With increasing n, the denominator term is monotonically nonincreasing. Since all examples eventually appear in S, the denominator converges to a limit, say $d \le 1$. For each k, the inferred probability f_k is monotonically nondecreasing with increasing n, until f_k is inconsistent with a new example, in which case it falls to zero and stays there henceforth. In the limit, only the f_k's that are consistent with the sequence of presented examples have positive inferred probability $\mathbf{m}(k)/d$. By Theorem 3, since $\mathbf{m}(k) = \Theta(2^{-K(k)})$, the highest inferred probability is carried by the rule f_k with least Kolmogorov complexity among the remaining ones. Similar statements hold after each initial segment of n examples, $n = 1,2,...$.

Reasoning inductively, we transform the *a priori* probability according to Formula (8), inferring a new *posterior* probability by the evidence of each initial segment of examples. At each step, we can select the rule with the highest inferred probability, and in the limit we have selected the proper rule. At each step we predict the rule with the highest inferred probability. Reformulating, if we want to infer a language L using this procedure, then:

(a) The Bayesian *a posteriori* probability for the correct answer converges to $c\,2^{-l(p)}/d$, where p is the shortest program which the reference machine uses to simulate M_0, where M_0 is the smallest TM that accepts L. This correct answer will have the highest probability in the limit. That is, inferred probability distribution over the underlying machines converges to a highest probability for M_0 *in the limit*. In other words, after n steps for some n, all the machines smaller than M_0 violate some data pair in S, and M_0 is the choice forever after step n.

(b) It is interesting to notice that the *a posteriori* probability decreases monotonically until it converges to $c\,2^{-l(p)}/d$ for p the program with which U simulates M_0. Smaller machines are chosen first and then canceled because they violate some data.

Predicting extrapolation. If we want to infer $f(s)$, rather than f, given the sequence of examples S, then using formulas (2) and (5), the inferred probability that $f(s) = a$ is (denoting a string $s_1 s_2 \cdots s_n$ as $s_{1:n}$):

$$P(f(s) = a \mid f(s_i) = a_i, i = 1,..,n) = \frac{\sum\{\mathbf{m}(j): f_j(s_i) = a_i, i = 1,..,n, f_j(s) = a\}}{\sum\{\mathbf{m}(j): f_j(s_i) = a_i, i = 1,..,n\}} \tag{9}$$

The Gold paradigm of inductive inference in the sense as originally studied by Gold in [Gold Language identification 1967] [Gold 1978 can be viewed simply as a computable approximation to Equation (8). The fundamental idea of the Gold paradigm is the idea called *identification in the limit* and a universal method of implementing the identification in the limit is called 'identification by enumeration'. These are contained in facts (a) and (b), as a computable analogue of Solomonoff's approach. We now investigate the correspondence between these two basic ideas in some detail.

Identification in the Limit views inductive inference as an infinite process. Formally, let M be an inductive inference method in order to derive some unknown rule R. If M receives a larger and larger set of examples (bigger and bigger initial segment S), a larger and larger sequence of M's conjectures is generated, say, $f_1, f_2, f_3, \cdots$. If there is some integer m such that f_m is a correct description of R and for all $n > m$

$$f_m = f_n,$$

then M *identified* R in the limit. Two facts deserve mentioning: M cannot determine whether it has converged and therefore stop with a correct hypothesis. M may be viewed as learning more and more information about the unknown rule R and monotonically increasing its approximation to R until the correct identification. Gold gave the best explanation to his definition:

> I wish to construct a precise model for the intuitive notion "able to speak a language" in order to be able to investigate theoretically how it can be achieved artificially. Since we cannot write down the rules of English which we require one to know before we say he can "speak English", an artificial intelligence which is designed to speak English will have to learn its rules from implicit information. That is, its information will consist of examples of the use of English and/or of an informant who can state whether a given usage satisfies certain rules of English, but cannot state these rules explicitly.
>
> $\cdots$ A person does not know when he is speaking a language correctly; there is always the possibility that he will find that his grammar contains an error. But we can guarantee that a child will eventually learn a natural language, even if it will not know when it is correct.

Identification by enumeration is a method to implement identification in the limit. It refers to the following guessing rule: Enumerate the class of rules in rule space. At step t, guess the unknown rule to be the first rule of the enumeration which agrees with data received so far. Formally speaking, in our setting, if we have received an initial segment S, then, given s, predict as the next example $(s, f(s))$ for f is the first rule that is consistent with S. Now if this can be done *effectively*, identification in the limit will be achieved. We say an induction method identifies a rule correctly in k steps if it will never produce wrong hypothesis starting from step k*. Let G and G' be two guessing methods. G will be said to be *uniformly faster* than G' if the following two conditions hold: (1) Given any R from the rule space, G will identify R correctly at least as soon as G', expressed in the number of examples needed, for all sequences of examples; and (2) for some R, some sequence of examples, G will identify R sooner than G'. Say a guessing method G is *optimal* if for any other guessing method G' there is a constant c such that: if R appears to be the ith rule in the enumeration of G', then it appears no later than ci in the enumeration of G. It is easy to prove that the identification-by-enumeration method will identify a hypothesis in the limit if this hypothesis can be identified in the limit at all. Further if G_0 is an identification-by-enumeration guessing rule, then there is no guessing rule uniformly faster than G_0. But only the Solomonoff procedure is optimal. Indeed,

Theorem 4. (a) *Identification-by-enumeration is a computable approximation to inductive inference (Solomonoff's inference) associated with Formula (8). (b) Neither method is uniformly faster than the other. (c) Solomonoff procedure is optimal, while identification-by-enumeration is not.*

Proof. (a) An effective enumeration for the identification-by-enumeration method, can be viewed as a computable approximation to Solomonoff's procedure according to formula (8) as follows. Let the effective enumeration of the rule space be: $R_1, R_2, R_3 \cdots$. Convert this to an effective prefix-free description of each rule R_i in the rule space. For instance, if $x = x_1, ..., x_n$ is a binary string, then $\bar{x} = x_1 0 x_2 0 ... 0 x_n 1$ is a *prefix-code* for the x's. Similarly, $x' = \overline{l(x)} x$ is a prefix-code. Note that $l(x') = l(x) + 2 \log l(x)$. We encode each rule R_i (a binary string) as $p\, i'$, where p is a (prefix) program that enumerates the rule space. The resulting code for the R_i's is an effective prefix-code. Denoting the length of the description of R_i by $|R_i|$, we have:

(i) if $i<j$, then $|R_i| \le |R_j|$; and

(ii) $\sum_i 2^{-|R_i|} \le 1$ (by Kraft's inequality).

Assign *a priori* probability $P(R_i) = 2^{-|R_i|}$ to rule R_i, $i = 1, 2, ...$ (This is possible because of (ii).) Using Formula (8) with $P(R_i)$ instead of $\mathbf{m}(i)$ yields a computable approximation to Solomonoff's inductive inference procedure. Formula (8) chooses the shortest encoded

* Although here we treat only the case when the procedure converges to one single rule, this definition allows that the procedure vacillate between the correct rules. Such definition is needed when, say, function f is not computable. Such research has been initiated by J. Case [case power of vacillation

consistent rule which coincides with the first consistent rule in the effective enumeration. This shows that identification by enumeration can be formulated as an computable approximation to Solomonoff's procedure.

We now show that neither method is uniformly faster than the other. Let $G_1, G_2, \ldots$ be an effective enumeration of the hypotheses space by a Gold procedure, and let $H_1, H_2, \ldots$ be the (noneffective) enumeration of the hypotheses space by decreasing *a priori* probability according to Solomonoff. In other words, $K(H_1) \le K(H_2) \le \cdots$. In both cases we deal with identification-by-enumeration, so it is known that there is no guessing rule uniformly faster than either of them.

To prove (c), let an arbitrary procedure G using identification-by-enumeration effectively enumerates rules in our rule space as $R_1, R_2, \cdots$. Then obviously $K(R_i) = K(i) + c_G$, where c_G depends on G. Hence $\mathbf{m}(R_i)$ is approximately at least $\frac{1}{c \cdot i}$ for some c. Hence the number of rules that have probability greater than this is at most $c \cdot i$. $\square$

Remark. What about non-uniform speed comparison? In case the particular rule f to be inferred is sufficiently *simple* (has low Kolmogorov complexity) then Solomonoff's procedure can be much faster than Gold's enumeration. Let f be the function we want to infer, and let $f = f_m$, with m minimal, in Gold's enumeration order. Let also $f = f_n$, for n with $K(n)$ minimal. To infer the correct f, in Gold's approach we must eliminate all f_k with $k < m$. But in Solomonoff's approach, we only need to eliminate all f_k with $K(k) < K(n)$. Now necessarily there are many f's that are 'simple' in the sense that $K(n) \ll l(m)$, for which e.g. Solomonoff's procedure works much (sometimes noncomputably) faster than Gold's method.

The following theorem sets limits on the number of examples needed to infer a particular function f.

Theorem 5. *Let $f_1, f_2, \ldots$ be an effective enumeration of the rule space. Suppose we want to infer $f = f_i$, with i minimal, from a set of n examples S as in (7). Let c be an appropriate large enough constant.*

(a) If $K(i) > K(f(s_1) \ldots f(s_n) \mid s_1 \ldots s_n) - c$, then it is impossible by any effective deterministic procedure to infer f correctly.

(b) If we can infer f correctly by computable approximation to Solomonoff's method (8) using only S, and c extra bits of information, then $K(i \mid S) \le c$.

(c) If $K(i \mid S) \le c$ then we can compute f_i from S and c bits extra information.

Proof. (a) Otherwise we would be able to compute i from a program of length significantly shorter than $K(i)$: contradiction. Items (b) and (c) are obvious. □

There is an enormous amount of research in the area under the title of Gold paradigm. We refer the readers to the articles [Angluin Smith] [Case Smith and the book [Osherson We present three examples of reasoning by means of Gold's paradigm in order to give a flavor of this research direction.

Example.[Gold, 1967] We can learn a function in the set of primitive recursive functions.

Proof. Effectively enumerate the set of all primitive recursive functions by $\psi_1, \psi_2, \cdots$. On any initial input segment $(x_1,y_1) \cdots (x_k,y_k)$, our inference machine just prints the least i such that ψ_i is consistent with the input, *i.e.*, $\psi_i(x_k)=y_k$ for $k=1, \cdots, n$.

Example. [Gold, 1967] We cannot learn in general a function in the set of all total recursive functions.

Proof. By diagonalization. Suppose M can identify all recursive functions. But then one can define a recursive function f so that the guesses of M will be wrong on f infinitely often. We construct f by simply simulating M. Let $f(0)=0$, Suppose the value of $f(0), f(1), \cdots, f(n-1)$ have been constructed. On input n, simulate M on initial input $f(0), f(1), \cdots, f(n-1)$. Then define $f(n)$ equal 1 plus the guess of M (modulo 2). So M never guesses f correctly. □

Example. One of the first studied problem was extrapolating a sequence. A machine M *extrapolates* a sequence $f(1), f(2), \cdots$ as follows. It makes an initial guess $f'(0)$. Then it inputs the real $f(0)$. At step i, based on previous inputs $f(1), f(2), \cdots, f(i-1)$, it guesses $f'(i)$. If there is a i_0 such that for all $i>i_0$ $f'(i)=f(i)$, then we say M *extrapolates* f. Bringing everything in our setting, the initial segment before step i is a sequence of pairs $(1,f(1))(2,f(2))...(i-1,f(i-1))$, and M extrapolates with the pair $(i,f'(i))$. It is not surprising that the class of functions computable by a Turing machine running in time $t(n)$, for any computable function t, can be extrapolated (by identification by enumeration).

4. Rissanen's Minimum Description Length Principle

Solomonoff's ideas about inductive reasoning have explicitly served as guiding principle in Rissanen's development of Minimum description length (MDL) principle. Let us derive Rissanen's MDL principle from Solomonoff's induction principle. For simplicity, we deal with only non-adaptive models. A non-adaptive model is a model $P(D \mid \theta)$ where the parameter vector $\theta=\theta(D)$ is estimated from n observed data points denoted by D.

Scientists formulate their theories in two steps: firstly a scientist must, based on scientific observations or given data, formulate alternative hypotheses, and secondly he selects one definite hypothesis. This is the subject of inference in statistics. Statisticians have developed many different principles to do this, like Occam's razor principle, the Maximum Likelihood principle, various ways of using Bayesian formula with different prior distributions. No single principle turned out to be satisfactory in all situations. Philosophically speaking, Solomonoff's approach presents an ideal way of solving induction problems using Bayes' rule with the universal prior distribution. However, due to the non-computability of the universal prior function, such a theory cannot be directly used in practice. Some approximation is needed in the real world applications. Further, from theory to inductive inference and statistical practice, there is still a big distance, for example, concrete formulae are needed.

Gold's principle was a particularly simple approximation to Solomonoff's induction - the sophisticated notion of probability distribution is replaced by linear enumeration. Now we will closely follow Solomonoff's ideas, but substitute a 'good' computable approximation to $\mathbf{m}(x)$. This results in Rissanen's **Minimum Description Length Principle** (MDL principle). He not only gives the principle, more importantly he also gives the detailed formulas on how to use this principle. This makes it possible to use the MDL principle in real problems. The principle can be intuitively stated as follows:

Minimum Description Length Principle. *The best theory to explain a set of data is the one which minimizes the sum of*

(1) *the length, in bits, of the description of the theory;*

(2) *the length, in bits, of data when encoded with the help of the theory.*

We now develop this MDL principle from Bayes' rule, Formula (1), using the Universal distribution $\mathbf{m}(x)$. Recall Bayes' formula:

$$P(H \mid D) = \frac{P(D \mid H) P(H)}{P(D)}.$$

Here H is an hypothesis, here a probability distribution, which we assume to be computable or anyway semicomputable, and D the observed data. We must choose the hypothesis H such that $P(H \mid D)$ is maximized. First we take the negative logarithm on both sides of the formula:

$$-\log P(H \mid D) = -\log P(D \mid H) - \log P(H) + \log P(D).$$

Since $P(D)$ can be considered as a normalizing factor, we ignore it in the following discussion.

Since we are only concerned with maximizing the term $P(H \mid D)$ or, equivalently, *minimizing* the term $-\log P(H \mid D)$, this is equivalent to minimizing

$$-\log P(D \mid H) - \log P(H).$$

Now to get the minimum description length principle, we only need to explain the above two terms in the sum properly. According to Solomonoff, when P is semicomputable, then we approximate P by $\mathbf{m}$. The prior probability $P(H)$ is set to $\mathbf{m}(H)=2^{-K(H)\pm O(1)}$, where $K(H)$ is the prefix Kolmogorov complexity of H. That is, $-\log P(H)$ is precisely the *length* of a minimum *prefix code*, or program, of the hypothesis H.

Similar argument applies to term $-\log P(D \mid H)$. Assuming P is semicomputable, using the conditional version of (3) and (4), we know that the universal semimeasure $\mathbf{m}(x)$ has the following properties.

(a) There is a constant c, such that $\mathbf{m}(D \mid H) \geq cP(D \mid H)$.

(b) The P-probability that $\mathbf{m}(D \mid H) \leq kP(D \mid H)$ is at least $1 - 1/k$.

By a conditional version of Theorem 3, $\mathbf{m}(D \mid H) = 2^{-K(D \mid H)\pm O(1)}$. Hence again $2^{-K(D \mid H)}$ is a reasonable approximation of $P(D \mid H)$, and minimizing $-\log P(D \mid H)$ can be considered as minimizing $K(D \mid H)$, *i.e.*, finding an H such that the description length, or the Kolmogorov complexity, of D given H is minimized. The term $-\log P(D \mid H)$ can also be thought as the *ideal code length* for describing data D, given hypothesis H. Such prefix code length can be achieved by the Shannon-Fano code. The term $-\log P(D \mid H)$, also known as the *self-information*, in information theory, and the negative log likelihood in statistics, can now be regarded as the number of bits it takes to redescribe or encode D with an ideal code relative to H.

In the original Solomonoff approach, H in general is a Turing machine. In practice we must avoid such an overly general approach in order to keep things computable. In different applications, the hypothesis H can mean many different things. For example, if we infer decision trees, H is a decision tree; In case of learning finite automata, H can be a finite automaton; In case we are interested in learning Boolean formulae, then H may be a Boolean formula; If we are fitting a polynomial to a set of points, then H may be a polynomial of some degree; In general statistical applications, one assumes that H is some model $H(\theta)$ with a set of parameters $\theta=\{\theta_1, \cdots, \theta_k\}$, where the number k may vary and influence the (descriptional) complexity of $H(\theta)$. In such case, from

$$-\log P(D \mid \theta) - \log P(\theta),$$

using Taylor expansion at the point of optimal $\hat{\theta}$ (for best maximum likelihood estimator), and taking only dominant terms, Rissanen has derived a formula for the minimum description length as

$$\min_{\theta,k}\{-\log P(D\,|\,\theta)+\frac{1}{2}k\log n\},$$

where k is the number of parameters in $\theta=\{\theta_1, \cdots, \theta_k\}$, and n is number of observations (or data points) contained in D. At the optimal k and θ, the term $1/2\,k\log n$ is called the optimum model cost.

Since $K(H)$ is not computable and hard to approximate, Rissanen suggested the following approach. First convert (or encode) H to a positive integer in $N=\{1,2,\cdots\}$. Then we try to assign prior distribution to each integer in N. Jeffreys [Jeffreys Theory of Probability suggested to assign probability $1/n$ to integer n. But this results an improper distribution since the series $\sum 1/n$ diverges. We modify Jeffreys distribution. It is possible, by iterating the idea of encoding n (viewed as the corresponding nth binary string) as $n' = \overline{l(n)}\,n$, to obtain a prefix-code such that $L(n)$ denotes the length of the code for n, with $L(n)$ defined by

$$l^*(n) = \log n + \log\log n + \cdots,$$

all positive terms, and

$$L(n) = l^*(n) + \log c,$$

where $c=2.865064\cdots$. Viz, it can be proved [Rissanen Universal prior integers that:

$$\sum_{n=1}^{\infty} 2^{-l^*(n)} = c.$$

Therefore, the existence of a prefix-code as claimed follows from Kraft's Inequality.

Assign prior probability $P(n)=2^{-L(n)}$ to each integer n. We obtain the following desired properties: (a) $\sum_{n=1}^{\infty}2^{-L(n)}=1$; and (b) integers n are coded by a prefix code. Hence, descriptions of two integers, n_1 and n_2, can be just concatenated to produce the code for the pair (n_1,n_2), and so on. The decoding process is trivial.

Using the MDL principle, Wax and Rissanen (according to Wax) and Quinlan and Rivest [Quinlan Rivest have developed procedures to infer decision trees. Other work by Wax [Wax Detection coherent and by Gao and Li [gao li applied MDL principle to recognition problems.

Example. We sketch an initial experiment we [gao li have performed in on-line handwritten character learning using the MDL principle. Inputing Chinese characters into computers is a difficult task. There are at least 5,000 characters in daily use, all of different shapes. Many methods have been invented for key-board input. Some have been successful in the limited sense that they can be used by trained typists only. Handwriting input is an alternative choice. Many such systems have been built with various recognition rates.

We [gao li have implemented such a system that learns handwritten characters from examples under the guidance of the MDL principle. We now sketch a simple experiment we have performed. An input character is drawn on a digitizer board with 200/inch resolution in both horizontal and vertical directions. The system learns a character from examples. The basic algorithm involves low level preprocessing, scaling, forming a prototype of a character (for learning), elastic matching (for recognizing), and so on. At the stage of forming a prototype of a character, we have to decide on the *feature extraction intervals*. Then we code a character into a prototype so that future inputs are classified according to their (elastic Hamming) distance to the prototypes.

Handwritten characters are usually quite arbitrary and prone to lots of noise. If the feature extraction interval is very small, then the algorithm will be very sensitive to errors and slight changes in the recognition phase, causing low recognition rate. If the feature extraction interval is very large, then it becomes less likely that we extract the essential features of a character and hence we get a low recognition rate again. We must compromise. The compromise is on the basis of minimum description length of prototypes.

We proceeded as follows to establish an optimal feature selection interval. A set of 186 characters drawings by one subject, exactly 3 examples for each of the 62 alphanumerical characters, were recorded. The character drawings were stored in a standardized integer coordinate system ranged from 0 to 30 in both x and y directions. These character drawings were then input to the system to establish a knowledge base, which formed the collection of prototypes with normalized real coordinates, based on some selected feature extraction interval. After the construction of knowledge base was finished, the system was tested by having it classify the same set of character drawings. If a character is misclassified, it is encoded using extra bits (i.e., the term $P(D \mid H)$). The *error code length* is the sum of the total number of points for all the incorrectly classified character drawings. The *model code* length is the total number of points in all the prototypes in the machine's knowledge base multiplied by 2. The factor of 2 comes from the fact that the prototype coordinates are stored as real numbers. This takes twice as much memory (in C) as the character drawing coordinates which are in integer form. The prototype coordinates are real instead of integer numbers, to facilitate the elastic matching process to give small resolution for comparisons of classification.

Thus, both the model code length and the error code length are directly related to the feature extraction interval. The smaller this interval, the more complex the prototypes, but the smaller the error code length. The effect is reversed if the feature extraction interval goes toward larger values. Since the *total code length* is the sum of the two code lengths, there should be a value of the feature extraction interval gives a minimum for the total code length. This feature extraction interval is considered to be the ‘best’ one in the spirit of the MDL principle. The corresponding model, or knowledge base, is considered to be optimal in the sense that it contains enough of the essence of the raw data but eliminates most redundancy of the noise component from the raw data. This optimal feature extraction interval can be found empirically by carrying out the above described build-and-test procedure repeatedly. That is, build the knowledge base, and then test it based on the same set of characters for which it was built. Repeat this for a number of different extraction intervals.

In fact, this actual optimization process is implemented on the system and is available whenever the user wants to call it. For our particular set of characters, the results

Figure 3. Optimization

of this optimization are given in Figure 3, which depicts three quantities: the model code length, the error code length, and the total code length versus feature extraction interval (SAMPLING INTERVAL in the Figure). For larger feature extraction intervals, the model code length is small but most of the character drawings are misclassified, giving a very large total code length. On the other hand, when the feature extraction interval is at the small end of the scale, all the training characters get correctly classified, and the error code length is zero. However the model code length reaches its largest value, resulting in a larger total code length again. The minimum code length occurred at extraction interval of 8, which

Figure 4. Optimization correct ratio

gives 98.2 percent correct classification. Figure 4 illustrates the fraction of correctly classified character drawings for the training data.

Whether the resulting ‘optimal’ model really performs better than the models in the same class, the knowledge bases established using different feature extraction intervals, is subject to testing it on new character drawings. For this purpose, the set of 62 handwritten characters were drawn again by the same person who provided the initial data to build the knowledge base. Thus the new data can be considered to be from the same source as the previous data set. The new data were classified by the system using the knowledge base built from the former data set of 186 character drawings, based on different feature extraction intervals. The testing result is plotted in Figure 5 in terms of the fraction of correct classification (CORRECT RATIO) versus feature extraction interval (SAMPLING INTERVAL). It is interesting to see that 100% correct

Figure 5. Test result

classification occurred at feature extraction intervals 5, 6 and 7. These values of feature extraction intervals are close to the optimized value 8. At the low end of feature extraction interval scale the correct classification drops, indicating disturbance caused by too much redundancy in the model. The recommended working feature extraction interval is thus either 7 or 8 for this particular type of character drawings. For more information on this research, see [gao li (preprint available from the first author).

5. Fisher's Maximum Likelihood Principle

Rissanen [universal prior has argued that Fisher's maximum likelihood principle [Fisher] [Gauss is a special case of the MDL principle. By our treatment of MDL it is therefore a more restricted computable approximation to Solomonoff's induction. The *Maximum Likelihood* principle says that given data D, one should use the hypothesis H that maximizes $P(D \mid H)$ or, equivalently, minimizes $-\log P(D \mid H)$, the first term in of the MDL principle. We will use H and θ interchangeably because θ is used by statisticians. What makes ML principle sound in statistics is the implicit assumption that each hypothesis H consists of a probability distribution $\theta=(\theta_1, \cdots, \theta_k)$ with the same number k of parameters, each parameter θ_i with fixed precision. In other words, in the probability distribution $P(D \mid H = \theta)$ the number k of parameters of θ, and the precision of each of them, is the same for each H. Hence, one assumes that the descriptions of *all* *hypotheses (models* θ) are of equal length; that is, the complexity of the models is considered to be fixed. This is, obviously, a *subjective* assumption. In contrast, the MDL principle minimizes the sum of $-\log P(D \mid H)$ and $-\log P(H)$ Intuitively, if one increases the description length of the hypothesis H, it may fit the data better and therefore decrease the description of data given H. In the extreme case, if one encodes all the data information into the model H precisely, $P(H)$ is minimized and $-\log P(H)$ is maximized. In that case, no code is needed to describe the data; that is, $P(D \mid H)$ is maximized (equals 1) and $-\log P(D \mid H)$ is minimized (equals 0).

On the other hand, if one decreases the description length of H, then this may be penalized by the increasing description length of the data, given H. In the extreme case say, H is a trivial hypothesis that contains nothing, then one needs 0 bits to describe H. But then, one gains no insight of data and has to 'plainly' describe the data without help from any hypothesis.

Hence one may consider the MDL principle as a more general principle than the ML principle in the sense that it considers the trade-off between the complexity of the model H and the power of the model to describe the data D, whereas the ML principle does not take the hypothesis complexity into account.

Yet the rationale behind the ML principle was to be objective by avoiding the 'subjective' assumption of the prior probability. The ML principle is equivalent with selecting the probabilistic model $P(D \mid \theta)$ which permits the shortest ideal code length for the observed sequence, provided that the model used in the encoding, *i.e.*, the parameter θ is given, too. Thus, the ML principle is just a special case of the MDL principle under the assumption that hypotheses are equally likely and the number of parameters in θ are fixed and small (so they do not make $P(D \mid \theta) = 1$). The shortcoming of the ML principle is that it cannot handle the situation where we do not know the number (and precision) of the parameters. For example, in the fitting polynomial example, the ML principle does not work well when the degree of the polynomial is not fixed. On the other hand the MDL principle works naturally for this example.

6. Jaynes' Maximum Entropy Principle

Rissanen [universal prior and M. Feder [Feder have shown that Jaynes' Maximum Entropy (ME) principle [Jaynes rationale maximum entropy] [Jaynes information inference] [Jaynes Probabilities can also be considered as a special case of the MDL principle. This is interesting since it is known in statistics that there are a number of important applications where the ML principle fails but where the maximum entropy formalism has been successful, and vice versa. In order to apply Bayes' theorem, we need to decide what the prior probability $p_i = P(H_i)$ is subject to condition

$$\sum_i p_i = 1,$$

and certain other constraints provided by empirical data or considerations of symmetry, probabilistic laws, and so on. Usually these constraints are not sufficient to determine the p_i's uniquely. Jaynes proposed to use the estimated values $\hat{p}_i$ which satisfy said constraints and maximize the entropy function

$$H = -\sum_i p_i \ln p_i$$

subject to the constraints. This is called the *maximum entropy (ME)* principle.

We now demonstrate the rationale behind the ME principle, its use, and its connection with the MDL principle following discussions in [Jaynes rationale] [Feder] [Rissanen universal prior Consider a random experiment with k possible outcomes in each trial, thus k^n possible outcomes in n trails. Let n_i be the number of times the ith value appears in an outcome D of n trials. Let frequency $f_i = n_i / n$, $i = 1, 2, ..., k$. The *entropy* of outcome D is:

$$H(f_1, \ldots, f_k) = -\sum_{i=1}^{k} f_i \ln f_i . \tag{10}$$

Let there be $m < k$ linearly independent constraints of the form

$$\sum_{i=1}^{k} a_{ji} f_i = d_j, \quad 1 \le j \le m, \text{ and} \tag{11}$$

$$\sum_{i=1}^{k} f_i = 1 \tag{12}$$

where the set $\mathbf{D} = \{d_1, ..., d_m\}$ is related to the observed data, measuring as it were m 'physical quantities' subject to the matrix $A = \{a_{ji}\}$.

Example. Consider a loaded die, $k = 6$. If we do not have any information about the die, then using the Epicurus' multiple explanation principle, we may assume that $p_i = 1/6$ for $i = 1, ..., 6$. This actually coincides with the ME principle, since $H(p_1, ..., p_6) = \sum_{i=1}^{6} p_i \ln p_i$ subject to (12) achieves maximum value $\ln 6 = 1.7917595$ for $p_i = 1/6$ for all i. Now suppose some experiments on the die have been performed, and it is observed that the die is biased and the average throw gives 4.5. That is,

$$\sum_{i=1}^{6} i\, p_i = 4.5.$$

In terms of Equation (11), we have $m = 1$, $\mathbf{D} = \{4.5\}$, and $a_{j1} = (1,2,3,4,5,6)$. Maximizing the expression in Equation (10), subject to constraints (11) and (12) gives estimates:

$$\hat{p}_i = e^{-\lambda i} (\sum e^{-\lambda i})^{-1}, \; 1 = 1,...,6,$$

where $\lambda = -0.37105$. Hence $(\hat{p}_1,...,\hat{p}_6) =$ $(0.0543, 0.0788, 0{,}1142, 0.1654, 0.2398, 0.3475)$. The maximized entropy $H(\hat{p}_1, \cdots, \hat{p}_6)$ equals 1.61358. How dependable is the ME principle? Jaynes has proven an 'entropy concentration theorem' which, for example, implies that in an experiment of $N = 1000$ trails, 99.99% of all outcomes satisfying the constraints of Equations (11) and (12) have entropy

$$1.602 \le H(\frac{n_1}{n}, \ldots, \frac{n_6}{n}) \le 1.614.$$

Now we turn to the MDL principle to deal with the same problem. The following argument can be derived from probabilistic assumptions. But Kolmogorov [Kolmogorov logical foundations 1969] [Kolmogorov Combinatorial foundations 1983 advocated a purely combinatorial approach, such as we give below, which does not need any such assumptions. Let $\theta = (p_1, \cdots, p_k)$ be the actual prior distribution of a random variable. We perform a sequence of n independent trials. Kolmogorov observed that the real substance of Formula (10) is that we need approximately $n\,H(\theta)$ bits to record the sequence of n outcomes. Namely, it suffices to state that each outcome appeared $n_1,...,n_k$ times, respectively, and afterwards give the index of which one of the

$$C(n_1, \cdots, n_k) = \frac{n!}{n_1! \cdots n_k!}$$

possible sequences D of n outcomes actually took place. For this no more than

$$k \log n + \log C(n_1, \cdots, n_k) + O(\log\log n)$$

bits are needed. The first term corresponds to $-\log P(\theta)$, the second term corresponds to $-\log P(D \mid \theta)$, and the third term represents the cost of encoding separators between the individual items. Using Stirling's approximation for the factorial function, we find that for large n this is approximately

$$n\,(-\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}) = n\,H(\frac{n_1}{n}, \cdots, \frac{n_k}{n}).$$

Since k and n are fixed, the least upper bound on the minimum description length for an arbitrary sequence of n outcomes under certain given constraints $\mathbf{D}$ is found by maximizing the term $\log C(n_1,...,n_k)$ subject to said constraints. This is equivalent to maximizing the entropy function (10) under constraints $\mathbf{D}$. (Such constraints may be derived, for example, from the laws of large numbers: in case of independent experiments with a probability distribution θ, we have $n_i/n \sim p_i$, and we have a certain rate of convergence with certain probability.)

7. Valiant Style Deductive Learning

Can we make Gold-style learning feasible? According to commonly accepted views in the theory of computation, this means that the learning algorithm should run in polynomial time - and hence also use but a polynomial number of examples. The latter condition necessarily implies that not all examples in an infinite domain can turn up. Hence we need to assume a mechanism for making a selection of examples. A deterministic selection fixes the sequence of examples drawn in advance, hence we would like to assume that examples are drawn from some distribution. The idea in Gold's approach that an inference algorithm should work for all sequences of examples then translates to the idea that the learning algorithm should work for all distributions.

The second unavoidable modification of the common approach in statistical inference, or recursion theoretical learning, imposed by the feasability constraint, is as follows. In traditional inference we want to learn a concept precisely in the limit. The feasibility restriction to a polynomial algorithm precludes the precise learning of nontrivial concepts, and therefore we have to relax precision to within a certain error. This corresponds with natural learning, where it is important that learning is fast, and it suffices to learn approximately.

We have now arrived at Valiant's proposal: a learning theory, where one wants to learn a concept with high propability, in polynomial time, and a polynomial number of examples, within a certain error, under all distributions on the examples. The additional computational requirements are orthogonal to the usual concerns in inference, and result in a distinctly novel theory. However, there are at least two problems with it:

(1) Under all distributions, many concept classes, including some seemingly simple ones,

are not known to be polynomially learnable or known not to be polynomially learnable if $NP \neq RP$, although some concept classes are polynomially learnable under some fixed distribution.

(2) In real life situations, it is sometimes impossible to sample according to underlying distributions.

Item (1) is counterintuitive for a proposed theory of machine learning; in fact it shows that Valiant's initially proposed requirements for learning are too strong. In practice, we usually do not have to make such a general assumption. Due to this reason several authors have proposed to study Valiant learning under fixed distributions. Then some previously (polynomially) unlearnable classes become learnable. For instance, the class of $\mu-DNF$-formulae is polynomially learnable under the uniform distribution. However, the assumption of *any special* distribution is obviously too restrictive and not practically interesting.

7.1. A New Approach

In [Li SIAM Simple Concepts we proposed to study Valiant-style learning under *all simple distributions*, which properly include *all* computable and semi-computable distributions. This allows us to systematically develop a theory of learning for *simple concepts* that intuitively should be polynomially learnable. To stress this point: maybe it is too much to ask to be able to learn all finite automata fast (humans cannot either), but surely we ought to be able to learn a *sufficiently simple* finite automaton fast (as humans can). Previous approaches looked at syntactically described classes of concepts. We introduce the idea of the restriction of a syntactically described class of concepts to the concepts that are simple in the sense of having low Kolmogorov complexity. This will cover most intuitive notions of simplicity. Our other restriction, from distribution-free learning to simple-distribution-free learning is also not much of a restriction. Already the computable distributions include *all* distributions we have a name for, like the uniform distribution, normal distribution, geometric distribution, Poisson distribution - so the even wider class of simple distributions ought to cover everything practically interesting.

It is an integral part of the proposed approach to also deal with the problem of inability of sampling according to underlying distributions. In real life the samples are sometimes (or often) provided by some mechanical or artificial means or good-willed teachers, rather than provided according to its underlying distribution. Naturally the simpler examples are provided first. Consider a situation where a robot wants to learn but there is nobody around to provide it with examples according to the real distribution. Because it does not know the real distribution, the robot just has to generate its own examples according to its own (computable) distribution and do experiments to classify these examples. For example, in case of learning a finite state black box (with resetting mechanism and observable accepting/ rejecting behavior). So the sampling distribution and the real distribution may be quite different.

7.2. Definitions

Definition. (1) Let X be a set. A *concept* is a subset of X. A *concept class* is a set $C \subseteq 2^X$ of concepts. An *example of a concept* $c \in C$ is a pair (x,b) where $b=1$ if $x \in c$ and $b=0$ otherwise. A *sample* is a set of examples.

(2) Let $c \in C$ be the target concept and P be a distribution on X. Given *accuracy parameter* ε, and *confidence parameter* δ, a *learning algorithm* A draws a sample S of size $m_A(\varepsilon,\delta)$ according to P, and produces a *hypothesis* $h = h_A(S) \in C$.

(3) We say C is *learnable* if for some A in above, for every P and every $c \in C$,

$$Pr(P(h \Delta c) > \varepsilon) \leq \delta,$$

where Δ denotes the symmetric difference. In this case we say that C is (ε,δ)–*learnable*, or *pac-learnable (probably approximately correct*).

(4) C is *polynomially learnable* if A runs in polynomial time (and asks for polynomial number of examples) in $1/\delta$, $1/\varepsilon$, and the length of the concept to be learned.

Definition. A distribution $P(x)$ is *simple* if it is (multiplicatively) dominated by a semi-computable distribution $Q(x)$. That is, there is a constant c such that for all x,

$$cQ(x) \geq P(x).$$

The first question is how large the class of simple distributions is. It certainly includes all semi-computable distributions and hence all distributions in our statistics books. It can be shown that there is a non-semicomputable distribution which is simple, and that there is a distribution which is not simple.

7.3. Discrete Sample Space

First we deal with discrete sample spaces. We show that if a concept is polynomially learnable under this *single* distribution then it is polynomially learnable in Valiant's sense under *all* 'simple' distributions if we sample according to the 'universal' distribution. We also provide new non-trivial learning algorithms for several (old and new) classes of problems under our assumption. These classes were not known to be polynomially learnable under Valiant's more general assumption, some were even NP-complete. For example, the class of DNF's such that each monomial has Kolmogorov complexity $O(\log n)$, the class of k-reversible DFA of Kolmogorov complexity $O(\log n)$, and the class of k-term DNF are polynomially learnable under our assumptions. All these results hold for the appropriate polynomial time computable variants - perhaps bringing the approach in the practicable domain.

Definition. The learning algorithm *samples according to* $\mathbf{m}(x)$, if in the learning phase the algorithm draws random samples from $\mathbf{m}(x)$. (We can formalize this in different ways.) We obtain the following completeness result.

Theorem 6. *A concept class C is polynomially learnable under the universal distribution* $\mathbf{m}$, *iff it is polynomially learnable for each simple distribution P, provided the sample is drawn according to* $\mathbf{m}$.

Proof. $P(x)$ is dominated by some semi-computable distribution $Q(x)$. $Q(x)$ is in turn dominated by $\mathbf{m}(x)$. Hence, there is a constant $c > 0$ such that for all x,

$$c\mathbf{m}(x) \geq P(x)$$

Assume C is learnable (in time t) under distribution $\mathbf{m}(x)$. Then one can run the learning algorithms with error parameter ε / c in polynomial time. Let *err* be the set of strings that are misclassified by the learned concept. So with probability at least $1-\delta$

$$\sum_{x \in err} \mathbf{m}(x) \leq \varepsilon / c.$$

Hence

$$\sum_{x \in err} P(x) \leq c \sum_{x \in err} \mathbf{m}(x) \leq \varepsilon.$$

Hence if the underlying distribution is $P(x)$ rather than $\mathbf{m}(x)$, we are still guaranteed to 'pac-learn' C (in time t), if sampling according to $\mathbf{m}(x)$. This still requires that the learning algorithm has the required constant c as additional input. The argument can be improved so that this extra input can be dispensed with [Li SIAM Learning simple □

Since **m** assigns higher probabilities to simpler strings, one could suspect that after polynomially many examples, *all* simple strings are sampled and the strings that are left unsampled have only very low (inverse polynomial) probability. However, the next theorem shows that this is not the case.

Theorem 7. *Let S be a set of* n^c *samples drawn according to* **m** *. Then*

$$\sum_{x \notin S} \mathbf{m}(x) = \Omega\left(\frac{1}{(\log n)^2}\right).$$

Now let us consider polynomially computable distributions. Again, all textbook distributions we know are polynomially computable. Call a distribution *polynomial simple* if it is dominated by a polynomially computable distribution. In all of the discussion below all Kolmogorov complexity (including the related notion **m**) can be replaced by its polynomial bounded version.

7.3.1. Learning under m(x)

In the [Li 1989 Simple Concepts we gave an example of a class of simple concepts, $\log n$-DNF, which is polynomially learnable under the universal distribution, and hence in our sense under all simple distributions, and which is not known to be polynomially learnable in the general Valiant model. Here we present a class that was shown to be not polynomially learnable in Valiant's sense, unless P = NP, but which is polynomially learnable under $\mathbf{m}(x)$.

DNF stands for 'disjunctive normal form'. A DNF is any sum $m_1+m_2+...+m_r$ of monomials, where each monomial m_i is the product of some literals chosen from a universe $x_1, \ldots, x_n$ or their negations $\bar{x}_1, \ldots, \bar{x}_n$. A k-term DNF is a DNF consisting of at most k monomials. A monomial in a DNF is *monotone* if no variable in it is negated. In [Pitt Valiant 35 1989
it was shown that learning a monotone k-term DNF by k-term (or $2k$-term) DNF is NP-complete. (In contrast to k-DNF is a DNF where each monomial consists of at most k literals. Recall, that k-DNF is learnable in Valiant's sense [Valiant 1984 learnable)

Theorem 8. *Monotone k-term DNF is polynomially learnable by monotone k-term DNF while sampling under* **m**.

Proof (Sketch). Assume we are learning a monotone k-term DNF $f(x_1, \cdots, x_n) = m_1 + \cdots + m_k$, where m_i's are the k monotone monomials (terms) of f.

Learning Algorithm.

0. Draw a sample of $n^{k'}$ examples, $k'>k+1$. Set DNF $g := \emptyset$. (g is the DNF we will eventually output as approximation of f.)

1. Pick a positive example $a=(a_1, \cdots, a_n)$. Form a monotone term m such that m includes x_i if $a_i=1$.

2. *for* each positive example $a=(a_1,...,a_n)$ *do*: if $a_i=0$ and deleting x_i from m violates no negative examples, delete x_i from m.

3. Remove from the sample all positive examples which are implied by m. Set $g \leftarrow g+m$. If there are still positive examples left, then go to step 1, else halt and return g.

We show that the algorithm is correct. Let us write $m_i \subseteq m$ for two monotone monomials if all the variables that appear in m_i also appear in m. At step 1, the monomial m obviously implies no negative examples, since for some monomial m_i of f we must have $m_i \subseteq m$. Step 2 of the algorithm keeps deleting variables from m. If at any time for no monomial $m_i \in f$ holds $m_i \subseteq m$, then there exists a negative example that contains at most k 0's such that it satisfies m but no m_i of f. This negative example is of Kolmogorov complexity at most $k \log n$, hence by the Chernoff formulae (Section 2.1) with high probability it is contained in the sample. Hence at step 2, with high probability, there will be an m_i such that $m_i \subseteq m$. Hence we eventually find a correct m_i (precisely) with high probability. Then at step 3, we remove the positive examples implied by this m_i and continue on to find another term of f. The algorithm will eventually output $g=f$ with high probability by standard calculations. □

Remark. Notice that this is not an approximation algorithm like the one in [Li 1989 Simple Concepts
to learn $\log n$-DNF. This algorithm outputs the precise monotone formula with high probability.

7.4. Continous Sample Space

Secondly, we deal with continuous sample spaces. For example, the uniform distribution now is defined as $L(\Gamma_x) = 2^{-|x|}$, where Γ_x denotes the set of all one-way infinite binary strings starting with x. This is the Lebesgue measure on interval [0,1]. While for discrete sample spaces all concept classes are Valiant learnable (although not all are polynomially learnable), this is not the case for continuous sample spaces. We can define the notion of ‘simple’ semimeasure and that of universal semicomputable semimeasure, over a continuous sample space, and show that all concept classes are learnable over each simple semimeasure D iff they are learnable under the universal semimeasure. In contrast with the discrete case w.r.t. polynomial learning, here we do not need to require that the learning algorithm samples according to the universal measure. For details, see [Li 1989 Simple Concepts

8. Acknowledgement

We are grateful Ray Solomonoff, Leonid Levin, and Peter Gács for conversations about algorithmic complexity and inductive reasoning. We thank Mati Wax for useful discussions on the MDL principle; Leslie Valiant for suggesting handwritten character recognition as a viable application; and Qiong Gao for participating in the realization of it. Fahiem Bacchus, Danny Krizanc and John Tromp read and commented on the manuscript.