11-1995

On the Impact of Forgetting on Learning Machines

Rūsiņš Freivalds
University of Latvia

Efim Kinber
Sacred Heart University

Carl H. Smith
University of Maryland - College Park

Recommended Citation

Freivalds, R., Kinber, E., & Smith, C. H. (1995). On the impact of forgetting on learning machines. *Journal of the Association for Computing Machinery*, 42(6), 1146-1168.

On the Impact of Forgetting on Learning Machines, Journal of ACM, Vol. 42,No. 6,November1995,pp.1146-1168

Article *in* Journal of the ACM · January 1995

CITATIONS
0

READS
19

3 authors, including:

Rūsiņš Freivalds
University of Latvia
366 PUBLICATIONS **1,888** CITATIONS

SEE PROFILE

Efim Kinber
Sacred Heart University
135 PUBLICATIONS **1,151** CITATIONS

SEE PROFILE

On the Impact of Forgetting on Learning Machines

RŪSIŅŠ FREIVALDS

University of Latvia, Riga, Latvia

EFIM KINBER

Sacred Heart University, Fairfield, Connecticut

AND

CARL H. SMITH

University of Maryland, College Park, Maryland

Abstract. People tend not to have perfect memories when it comes to learning, or to anything else for that matter. Most formal studies of learning, however, assume a perfect memory. Some approaches have restricted the number of items that could be retained. We introduce a

---

This work was facilitated by an international agreement under National Science Foundation (NSF) Grant 91-19540.

Results collected in this paper were published in the proceedings of the following workshops and symposia: FREIVALDS, R., KINBER, E., AND SMITH, C. 1993a. Learning with a limited memory. In *Notes of the AAAI Spring Symposium on Training Issues in Incremental Learning*; FREIVALDS, R., KINBER, E., AND SMITH, C. 1993b. On the impact of forgetting on learning machine. In *Proceedings of the 6th Annual Workshop on Computational Learning Theory*. ACM, New York, pp. 165–174; FREIVALDS, R., KINBER, E., AND SMITH, C. 1993c. Probabilistic versus deterministic memory limited learning. In *Record of the Workshop on Algorithmic Learning for Knowledge Processing*; FREIVALDS, R., KINBER, E., AND SMITH, C. 1994. Quantifying the amount of relevant information. In *Notes of the AAAI Spring Symposium on Relevance*; and FREIVALDS, R., AND SMITH C. 1992. Memory limited inductive inference machines. In *Lecture Notes in Computer Science*, vol. 621. Springer-Verlag, New York, pp. 14–29.

The work of R. Freivalds was supported by the Latvian Council of Science grants No. 90.619 and 93.599.

The work of C. H. Smith was supported by NSF grants 90-20079 and 93-01339.

Authors' addresses: R. Freivalds, Institute of Mathematics and Computer Science, University of Latvia, Raina bulvāris 29, LV-1459, Riga, Latvia; E. Kinber, Department of Computer Science, Sacred Heart University, 5151 Poule Ave., Fairfield, CT 06432-1000; C. H. Smith, Department of Computer Science, University of Maryland, College Park, MD 20912.

complexity theoretic accounting of memory utilization by learning machines. In our new model, memory is measured in bits as a function of the size of the input. There is a hierarchy of learnability based on increasing memory allotment. The lower bound results are proved using an unusual combination of pumping and mutual recursion theorem arguments. For technical reasons, it was necessary to consider two types of memory: long and short term.

Categories and Subject Descriptors: F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*automata, relations among models, unbounded-action devices*; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation—*probabilistic computation*; F.1.3 [**Computation by Abstract Devices**]: Complexity Classes—*complexity hierarchies*; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*recursive function theory*; 1.2.6 [**Artificial Intelligence**]: Learning—*induction, concept learning*

General Terms: machine learning, memory limited learning, inductive inference, Kolomogorov complexity

Additional Key Words and Phrases: probabilistic automata, pumping lemma, recursion theorem

1. *Introduction*

Various aspects of machine learning have been under empirical investigation for quite some time [Michalski et al. 1983; Shapiro 1987]. More recently, theoretical studies have become popular [Haussler 1992; Fulk and Case 1993; Haussler and Pitt 1988; Rivest et al. 1989; Warmuth and Valiant 1991]. The research described in this paper contributes toward the goal of understanding how a computer can be programmed to learn by isolating features of incremental learning algorithms that theoretically enhance their learning potential. In particular, we examine the effects of imposing a limit on the amount of information that a learning algorithm can hold in its memory as it attempts to learn. While this idea in itself is not novel, our approach is. Our results clarify and refine previous attempts to formalize restricted memory learning.

In this work, we consider machines that learn programs for recursive (effectively computable) functions. This approach is very abstract. Many, if not all, of the implementation details are ignored, allowing a focus on fundamental issues. By choosing such a high level of abstraction, any hope of having any direct impact on the production and implementation of learning algorithms is lost. In return, we gain an intuition that is not predicated on any particular implementation strategy, computer architecture, or any other product of technology. Several authors have argued that such studies are general enough to include a wide array of learning situations [Angluin and Smith 1983, 1987; Blum and Blum 1975; Case and Smith 1983; Gold 1967; Osherson et al. 1986]. For example, a behavior to be learned can be modeled as a set of stimulus and response pairs. Assuming that any behavior associates only one response to each possible stimulus, behaviors can be viewed as *functions* from stimuli to responses. It is possible to encode every string of ASCII symbols in the natural numbers. These strings include arbitrarily long texts and are certainly sufficient to express both stimuli and responses. By using suitable encodings, the learning of functions represents several, ostensibly more robust, learning paradigms.

Hence, for the purpose of a mathematical treatment of learning, it suffices to consider only the learning of functions from natural numbers to natural numbers. A variety of models for learning recursive functions have been considered, each representing some different aspect of learning. The result of the learning will be a program that computes the function that the machine is trying to learn. Historically, these models are motivated by various aspects of

human learning [Gold 1967] and perspectives on the scientific method [Popper 1968].

We say that learning has taken place because the machines we consider must produce the resultant program after having ascertained only finitely much information about the behavior of the function. The models we use are all based on the model of Gold [1967] that was cast recursion theoretically in Blum and Blum [1975]. First, we briefly review the basics of the Gold model and then proceed to define the memory limited version of the basic model that will be investigated in this paper.

2. *The Gold Model*

People often hold steadfast beliefs that they later discover to be false. At various points in time, the scientific community was convinced that the earth was flat, the earth was the center of the universe, time is absolute, etc. Hence, one can never be absolutely sure that they have *finished* learning all there is to learn about some concept. We must always be prepared to embrace a better explanation of some phenomenon that we thought had been learned. Gold, in a seminal paper [Gold 1967], defined the notion called *identification in the limit*. This definition concerned learning by algorithmic devices now called *inductive inference machines* (IIMs). An IIM receives as input the range of a recursive function, an ordered pair at a time, and, while doing so, outputs computer programs. Since any IIM can buffer its input so as to process it in the natural domain increasing order, $f(0), f(1), \ldots$, we can assume without loss of generality that the input is received by an IIM in this natural order. However, such buffering may not be possible within the confines of the memory limitations we impose below (see Theorem 5.5). Primarily, we will calculate memory utilization for the natural domain increasing order only. In this way, we follow a long standing tradition and give a unifying theme to our results. An IIM, on input from a function f, will output a potentially infinite sequence of programs $p_0, p_1, \ldots$. The IIM *converges* if either the sequence is finite, say of length $n + 1$, or there is program p such that for all but finitely many i, $p_i = p$. In the former case, we say the IIM converges to p_n, and in the latter case, to p. In general, there is no effective way to tell when, and if, an IIM has converged.

Following Gold, we say that an IIM M *identifies* a function f in the limit (written: $f \in EX(M)$), if, when M is given the range of f as input in any order, it converges to a program p that computes f. If the IIM identifies some function f, then some form of learning must have taken place, since, by the properties of convergence, only finitely much of the range of f was known by the IIM at the (unknown) point of convergence. The terms, *infer* and *learn*, will be used as synonyms for identify. Each IIM will learn some set of recursive functions. The collection of all such sets, over the universe of effective algorithms viewed as IIMs, serves as a characterization of the learning power inherent in the Gold model. This collection is symbolically denoted by EX (for explanation) and is defined rigorously by $EX = \{U \mid \exists M(U \subseteq EX(M))\}$. Mathematically, this collection is set-theoretically compared with the collections that arise from the other models we discuss below. Many intuitions about machine learning have been gained by working with Gold's model and its derivatives. In the next section, we describe the variants of Gold's model that we examine in this paper.

3. *Limited Memory Learning*

Although different people learn at different rates, with varying degrees of success, very few of us learn with perfect recall of the data upon which the learning is based. This observation of human behavior has prompted several investigations of learning with less than perfect memory in a variety of disciplines. Within the field of neural modeling, it has been suggested that one of the functions of rapid eye movement (REM) sleep is to discard some memories to keep from overloading our neural networks [Crick and Mitchison 1983]. Independent simulations have verified that occasional "unlearning" aids in learning [Hopfield et al. 1983]. In a similar vein, neural networks with a limitation on the type of the weight in each node were considered in Siegelmann and Sontag [1992]. The types considered are integer, rational and real. Each successive type can, potentially, place higher demands on memory utilization within each node. Each type also expands the inherent capabilities of the neural networks using that type of node weights.

Linguists interested in how children learn language have hypothesized many mechanisms for remembering. Braine [1971] suggested that human memory is organized as a cascading sequence of memories. The idea is that items to be remembered are initially entered in the first level of the memory and then later moved to successive levels, finally reaching long-term memory. In Braine's model, each of the transitionary memory components are subject to degradations. Consequently, items to be remembered that are not reinforced by subsequent inputs may be eliminated from some level of the memory before they become permanently fixed in memory. Wexler and Culicover [1980] formalized many notions of language learning, including one where a device (essentially an inductive inference machine) was to learn having access to the most recently received data and the machines' own most recent conjecture. Their model was generalized in Osherson et al. [1986] to allow the learning mechanism access to the last n conjectures as well as the most recently received data item. This generalization was shown not to increase the potential of such mechanisms to learn languages.

A study of learning functions with a limited amount of memory was initiated by Wiehagen [1976]. He defined *iterative* strategies that, like the Wexler and Culicover model, had access to only the next data item and the current hypothesis. Also defined were *feedback* strategies that were allowed to remember the current hypothesis and a single, algorithmically selected, data item. Wiehagen showed that the iterative strategies were not as powerful as the feedback ones, and neither were as powerful as the unrestricted strategies of the Gold model. Furthermore, iterative learning, and hence feedback learning as well, are potentially more powerful than *consistent* learning where the inference machines are constrained to only output programs that are correct on all the data seen so far [Wiehagen 1976]. Miyahara generalized Wiehagen's work in two ways. Firstly, the iteratively working strategies were allowed to remember the n most recent conjectures. As was the case for learning languages, it makes no difference how many previous hypotheses are remembered for learning functions—one is enough [Miyahara 1987]. Furthermore, the hierarchies based on the number of admissible errors in the final answer, as discovered in Case and Smith [1983], were shown to hold for a variety of types of iterative learning. The strict inclusion of the class of sets learnable by iteratively working strategies in the similar class for all strategies (EX) was

shown to hold when anomalies are allowed [Miyahara 1989]. Jantke and Beick [1981] considered order-independent iterative strategies and showed that Wiehagen's results hold with order restrictions removed.

The conclusion reached in the above-mentioned work on learning functions was that restricting the data available to the inference machine also reduces its learning potential. A different approach to memory-limited learning was investigated in Heath et al. [1991]. The issue addressed in their work is to calculate how many *passes* through the data are needed in order to learn. In our model, the decision to retain data must be made when the data is first encountered. The models described below constrain the amount of what can be retained by an IIM, without placing any provisions on the content of the remembered data.

There have been other studies with similar motivations. For example, based on the observation that people do not have enough memory to learn an arbitrarily large grammar for a natural language, a study of learning minimal-size grammars was initiated [Case et al., to appear]. There has been a large body of work addressing the inference of minimal-size programs. See Freivalds [1990] for a survey.

There have been a few results concerning space limited learning in the PAC (probably approximately correct) model [Valiant 1984]. Haussler [1985] showed how to PAC learn strictly ordered decision trees using space linear in the size of the smallest possible decision tree. Boucheron and Sallantin [1988] showed that some classes of Boolean functions can be learned time efficiently using only logarithmic (in the number of variables) space. PAC learning while remembering only a fixed number of examples, each of a bounded size is considered in Ameur et al. [1993], Floyd [1989], and Hembold et al. [1989]. The most general investigation on this line was the observation in Schapire [1990] that the boosting algorithm can be made reasonably space efficient as well. Sample complexity gives only a very crude accounting of space utilization. Learning procedures may want to remember other information than just prior examples. For example, all algorithms are based on some underlying finite state device. The states of the underlying finite state machine can also be used as a form of long-term memory. Consequently, the sample complexity metric neglects to count some of the long-term storage employed by learning algorithms. Lin and Vitter [1994] consider memory requirements for learning sufficiently smooth distributions. Since they assume that the inputs are in some readable form, the issue of how much space it takes to store a number never arises.

We now describe the model investigated in this paper. To ensure an accurate accounting of the memory used by an IIM, we will henceforth assume that each IIM receives its input in such a way that it is impossible to back up and reread some input after another has been read. All of the previously mentioned models of learning languages or functions measured memory used in the number of data items or hypotheses that could be remembered. Since it is possible to encode an arbitrary finite set within any single hypothesis, coding techniques played a major role in the proofs of some of the above-mentioned results. Computers, and humans, use storage proportional to the size of what is being remembered. To circumvent the use of coding techniques, the memory used will be measured in trits (input alphabet consists of three elements), as opposed to integers. Each data entry will appear as a bit string with a designated delimiter separating the entries. The delimiter will be viewed as a "special bit" and we will henceforth count the memory utilization in bits.

Each of the machines we consider will have two types of memory. In the *long-term memory*, the IIM will remember portions of the input it has seen, prior conjectures, state information pertaining to the underlying finite state device and perhaps other information as well. In addition, each machine will have a potentially unlimited *short-term memory* that will be annihilated every time the IIM either outputs a new conjecture or begins reading the bits corresponding to another point in the graph of the mystery function providing the input to the IIM. The short-term memory clear operation is done automatically and takes one time step. The short-term memory is necessary to ensure an accurate accounting of the real long-term memory utilization of a learning process. It might be that some very space consuming computation must be performed in order to decide which few bits of information to retain and which to discard. Without a short-term memory, such a temporary use of space would artificially inflate the long-term memory needed by a learning algorithm.

As an added side benefit, we note that our technically motived model bears a strong resemblance to contemporary models of memory function with respect to the dichotomy between long- and short-term memory that was initiated in Miller [1956]. This dichotomy is evident in neural nets. The weights in the nodes correspond to long-term storage and the calculations as to how to update the weights is carried out using a short-term memory [Levine 1991]. Some well-known implementations of learning algorithms, the Soar project [Rosenbloom et al. 1991; Servan-Schrieber 1991] and the ACT* project [Anderson 1983], also divide memory into long- and short-term components. The Soar project uses the concept of "chunking" [Miller 1956] as a way to convert traces of problem solving behavior into new rules, freeing up space in the working short term memory in the process. The rules of Soar, in some sense, are analogous to the states of the finite state devices that we study. As in the Soar program, we keep state information in long-term memory. Another similarity between our model and the way Soar operates is that temporary calculations are lost in both schemes. When Soar reaches an impasse, some calculations are performed to generate a new subgoal. Just like our short-term memory, the subgoal calculations are lost when an appropriate one is found.

Under the above conventions concerning the use of long- and short-term memory, we proceed to define our limited memory model. We say that $U \subseteq EX(M) : g$ if g is a recursive function such that for any $f \in U$, $f \in EX(M)$ and M uses no more than $g(n)$ bits of long-term memory, where n is the number of bits of the range of f, from the natural domain increasing order enumeration, that M has observed. In one of our results, when we discuss the effects of changing the order of the input, we consider the input as arriving in ordered pairs. A more desirable model would be to consider inputs arriving in any order and to consider the worst possible case. In light of the Theorem 5.5 below, such a model would not be very robust. Consequently, in this preliminary study, we focus on the IIMs receiving their input in the traditional, increasing-domain-order enumeration. The collection of all sets of functions inferrible with memory bound, given by g, is denoted by $LEX : g$, where $LEX : g = \{U \mid \exists M(U \subseteq EX(M) : g\}$. To handle the important special case of the memory bound being a constant function, we will write $EX : c$ to denote

$$\bigcup_{g \in \mathscr{C}} EX : g$$

for $\mathscr{C}$ the collection of all constant functions.

A few more technical definitions are needed. Natural numbers ($\mathbb{N}$) will serve as names for programs. The function computed by program i will be denoted by φ_i. It is assumed that $\varphi_0, \varphi_1, \ldots$ forms an acceptable programming system [Machtey and Young 1978; Rogers 1967]. The quantifier $\overset{\infty}{\forall}$ is read "for all but finitely many." Sometimes, it will be convenient to represent a function by a sequence of values from its range. Such a representation is called a *string* representation. So, for example, the sequence $01^20^43^\infty$ represents the (total) function:

$$f(x) = \begin{cases} 0 & \text{if } x = 0 \text{ or } 3 \leq x \leq 6, \\ 1 & \text{if } 1 \leq x \leq 2 \\ 3 & \text{otherwise.} \end{cases}$$

This example function has two *blocks* of consecutive 0's, one of length 1 and the other of length 4. The string representation of a finite function is called a *fragment*. The length of any string α, in characters, is denoted by $|\alpha|$. A class U of recursive functions is dense iff for every finite function there is an $f \in U$ extending it. For example, the functions of finite support are a dense class.

4. *Preliminary Results*

In order to get a rough idea of the relative learning power of $EX:c$ type inference, we will employ the set U_0 of functions of finite support and the set U_1 of self-describing functions. These sets were introduced in Barzdins [1974] and Blum and Blum [1975] and used in Case and Smith [1983] and Freivalds and Smith [1993] to separate various classes of learnable sets of functions. Let $U_0 = \{f \mid f \text{ is recursive and } \overset{\infty}{\forall} x(f(x) = 0)\}$ and $U_1 = \{f \mid f \text{ is recursive and } \varphi_{f(0)} = f\}$.

PROPOSITION 4.1. $U_1 \in EX:c$.

PROOF. The IIM that, upon receipt of input pair $(x, f(x))$, compares x with 0. If there is a match, the value $f(x)$ is copied from the input register to the output register. The memory is needed only to store the constant 0. Clearly, $U_1 \in EX:c$. □

PROPOSITION 4.2. $U_0 \notin EX:c$.

PROOF. Suppose by way of contradiction that M is an IIM such that $U_0 \in EX:c(M)$. A pumping lemma type argument is used [Bar-Hillel et al. 1951] (see also Lewis and Papadimitriou [1981]). The string representation of functions is used in this proof. Since M has a constant, finite amount of memory, there are strings σ and τ such that

(1) σ and τ contain only 0's and 1's,
(2) τ contains at least one 1,
(3) M, on input $\sigma\tau$, outputs a conjecture while reading τ, and
(4) the contents of M's memory (including current state) is the same after reading $\sigma\tau$ as it was when it had just finished reading σ.

Such a σ and τ must exists, as we have assumed that M's memory is of a fixed size. By the choice of σ and τ, M's internal state, memory contents and most recent conjecture are identical after reading $\sigma\tau$ and $\sigma\tau^2$. Consequently, M cannot distinguish the (distinct) functions $\sigma\tau 0^\infty$ and $\sigma\tau^2 0^\infty$, both of which are in U_0. □

COROLLARY 4.3. $EX : c \subset EX$.

PROOF. By definition, $EX : c \subseteq EX$. The inclusion is proper by Proposition 4.2 and the fact that $U_0 \in EX$ [Blum and Blum 1975]. □

Another type of inference that may be relevant to neural networks is the class *PEX* defined in Case and Smith [1983] and studied in Case and NgoManguelle [to appear]. A set of functions U is in *PEX* just in case there is an IIM that outputs only programs for total recursive functions and $U \subseteq EX(M)$. The collection of sets *PEX* is defined analogously. In this case, the witnessing IIM M is called *Popperian.* Virtually all practical learning systems only consider hypotheses that correspond to total recursive functions.

COROLLARY 4.4. *$EX : c$ and PEX are incomparable.*

PROOF. U_0 is in *PEX* [Case and Smith 1983], therefore, by Proposition 4.2, $PEX - EX : c \neq \emptyset$. $U_1 \notin PEX$, hence, by Proposition 4.1, $EX : c - PEX \neq \emptyset$. □

For our final result in this section, we will verify that linear long-term memory is sufficient for any learning task. Based on this result, our ensuing discussion of memory-limited *EX* inference will be based on sublinear bounding functions.

THEOREM 4.5. *There exists a constant c such that if $g = \lambda x[x + c]$, then $EX = EX : g$.*

PROOF. Clearly, $EX : g \subseteq EX$. Suppose $S \in EX$ as witnessed by the IIM M. We describe the operation of an IIM M' such that $S \subseteq EX(M') : g$, where $g = \lambda x[x + c]$, and the constant c is to be chosen large enough to accommodate the state information of M'. Suppose $f \in S$. M' reads the graph of f for input and copies the information into long-term memory. The long-term memory bound of g is a large enough bound to accommodate all the input and the necessary state information. After M' completes reading the input bits corresponding to another element of the graph of f, M' then uses its short-term memory to mimic the behavior of M using the long-term memory of M' for input. M' then produces as output the last conjecture produced by M on the input in the long-term memory of M'. Since the short-term memory is destroyed every time M' produces a conjecture, the simulation of M must start again from the beginning each time. If M declines to produce an output on some initial segment of f such that no additional output is produced on the next longer initial segment of f, then M', on the longer segment, will repeat the previous output. So, aside from, perhaps, some extra repetitions of the most recent hypothesis, the sequence of conjectures produced by M' on f is the same as sequence of hypothesis produced by M on f. Hence, if $M(f)$ converges, then $M'(f)$ converges to the same value. Therefore, $S \subseteq EX(M') : g$. □

Notice that if $g = \lambda x[2 \cdot x]$ long-term memory were allowed, then the simulation described above could be performed in the part of long-term memory that was not used to save the input data. In this case, no short-term memory whatsoever would need to be used. Hence, the above result shows that any set in *EX* can be inferred using a linear amount of long-term memory and no short-term memory at all.

5. *Bounds on Long-Term Memory*

Our first result generalizes Proposition 4.2 to show that, not only is linear long-term memory sufficient for any learning task, sometimes it is necessary as well.

THEOREM 5.1. *There exists a constant c, a function $g = \lambda x[x + c]$, and class U of total recursive functions such that if h is any sublinear recursive function, then $U \in EX : g - EX : h$.*

PROOF. Let g and h be as in the hypothesis. Let U be the class of $\{0, 1\}$ valued functions of finite support. Since U is in EX, by Theorem 4.5 there exists a constant c and function $g = \lambda x[x + c]$ such that $U \in EX : g$. It remains to show that $U \notin EX : h$. Suppose, by way of contradiction, that $U \in EX(M) : h$ for some IIM M. We consider the natural, increasing domain order enumeration of the functions in U. Let n be such that $h(n) < n$. Notice that if $f \in U$ then $f(0), f(1), \ldots, f(n-1)$ can be represented by n bits. Consider all length n initial segments of the functions in U. There are 2^n such initial segments. The number of different memory configurations possible in $h(n) < n$ space is $2^{h(n)} < 2^n$. Hence, there are two different length n initial segments, say σ_0 and σ_1, that will result in the same memory configuration in M's long-term memory. Since there are infinitely many functions of finite support extending σ_0 and we have assumed that M can learn them all, there must be a τ_0, extending σ_0, such that $M(\sigma_0) \neq M(\tau_0)$ and the range of τ_0 is $\{0, 1\}$. Let m be the length of the initial segment τ_0. Define τ_1 as:

$$\tau_1(x) = \begin{cases} \sigma_1(x) & \text{if } x < n, \\ \tau_0(x) & \text{if } n \leq x < m. \end{cases}$$

Consider the functions f_0 and f_1 defined as follows:

$$f_0(x) = \begin{cases} \tau_0(x) & \text{if } x < m, \\ 0 & \text{otherwise} \end{cases}$$

$$f_1(x) = \begin{cases} \tau_1(x) & \text{if } x < m, \\ 0 & \text{otherwise} \end{cases}$$

M will be in the same long-term memory configuration after seeing the first n values of either f_0 or f_1. Both functions are identical after that point. Hence, M will exhibit the same limiting behavior on both f_0 and f_1, contradicting the assumption that $U \subseteq EX(M) : h$. □

Notice that for the class U of the proof of Theorem 5.1 for any $f \in U$, only a constant amount of long-term memory is needed. However, the constant changes for each function from U. The proof of Theorem 5.1 showed that at least linear long-term memory was needed at some point in the inference of any function from U. By using a more complicated argument, we can show that a linear amount of long term memory will be needed infinitely often when learning any function from some (more complicated) class.

We proceed to develop some techniques to prove lower bound results. Eventually, we will give a general theorem (Theorem 5.3). First, we give an interesting example class that we will show has logarithmic lower and upper bounds. Let U_{SQ} be the class of recursive functions f with prefixes of the form $1^n ij$, $i \neq 1$, where $\varphi_i = f$ if n is a perfect square and $\varphi_j = f$, otherwise.

THEOREM 5.2. *There exists a constant c such that $U_{SQ} \in EX: g$ where $g = \lambda x[x + c]$. Furthermore, if $U_{SQ} \in EX: h$ for any recursive function h with $h(x) \leq g(x)$ for all x, then $h(x) > 1/2 \log x$ for infinitely many x.*

PROOF. To see that $U_{SQ} \in EX: g$, notice that the initial prefix of 1's can be read and the value of n can be held in $g(n)$ bits. Use short-term memory to decide if n is a perfect square, leaving the answer in long-term memory. In another state, an IIM decides to output the next input, or the one after. The constant c is 1 plus the amount of memory needed for the state information of the IIM informally described above.

Suppose M is an IIM such that $U_{SQ} \in EX(M): h$. Consider giving M different length prefixes of all 1's. We claim that for any natural numbers a, b, and m such that $2^{2 \cdot m} \leq a < b < (2^m + 1) \cdot (2^m + 1)$ M will have different long-term memory contents after seeing 1^a and 1^b. Suppose by way of contradiction, that a and b satisfy the numeric constraints of the claim, yet M's long-term memory is the same after seeing the segments 1^a and 1^b. There are two cases to consider in order to verify the claim.

Case 1. $a = 2^{2 \cdot m}$. Then a is a perfect square and b is not. By the mutual recursion theorem [Smullyan 1961], there are programs e_1 and e_2 such that $\varphi_{e_1} = 1^a e_1 e_2 0^\infty$ and $\varphi_{e_2} = 1^b e_1 e_2 0^\infty$. Notice that both φ_{e_1} and φ_{e_2} are in U_{SQ}. Since M's long-term memory is assume to be unable to distinguish 1^a from 1^b, M will exhibit the same output behavior on both φ_{e_1} and φ_{e_2}. Hence, M fails to infer one of them, contradicting the assumption that $U_{SQ} \in EX(M): h$.

Case 2. $a > 2^{2 \cdot m}$. Then neither a nor b is a perfect square. Let $c = (2^m + 1) \cdot (2^m + 1) - b$. Hence, $b + c$ is a perfect square and $a + c$ is not. By the mutual recursion theorem there are programs e_1 and e_2 such that $\varphi_{e_1} = 1^{b+c} e_1 e_2 0^\infty$ and $\varphi_{e_2} = 1^{a+c} e_1 e_2 0^\infty$. Notice that both φ_{e_1} and φ_{e_2} are in U_{SQ}. Since M's long-term memory is assumed to be unable to distinguish 1^a from 1^b, it will also be unable to distinguish 1^{a+c} from 1^{b+c}. Therefore, M will exhibit the same output behavior on both φ_{e_1} and φ_{e_2}. Hence, M fails to infer one of them, contradicting the assumption that $U_{SQ} \in EX(M): h$.

As a consequence of the claim, M will have different long-term memory contents for each of the prefixes of 1's with lengths $2^{2 \cdot m}, 2^{2 \cdot m} + 1, 2^{2 \cdot m} + 2, \ldots (2^m + 1) \cdot (2^m + 1) - 1$. Since there are $2^{m+1} + 1$ different memory contents, at least one of these requires at least $m + 1$ bits to represent. Since each of the prefixes above has length at most the logarithm of $2m + 1$, we note that for the particular prefix in question, the amount of memory used is at least $m + 1 > (2m + 1)/2 > 1/2$ (length of the prefix). Since m was arbitrary, this shows that if $U_{SQ} \in EX(M): h$, then for infinitely many values of x, $h(x) > 1/2 \log x$. □

The next result that we prove shows that there is a hierarchy based on larger and larger amounts of long-term memory utilization. For the purposes of the following theorem, we will say that a recursive function f is *almost surjective* if there is an n such that $\{x \mid x \geq n\}$ is included in the range of f.

THEOREM 5.3. *Suppose g is a nondecreasing, almost surjective recursive function such that $g = O(n)$ and h is a recursive function such that $h = o(g)$. Then there is a class of recursive functions U such that $U \notin EX: h$, and $U \in EX: g$. Furthermore, if $U \in EX(M): f$, for some IIM M, then $f(n) > h(n)$ for infinitely many n.*

PROOF. Suppose g is a nondecreasing, almost surjective recursive function such that $g = O(n)$. Define s_g, the *pseudo-inverse* of g, as $s_g(n)$ is the least value x such that $g(x) \geq n$. Note that since g is almost surjective, such an x will exist for all n. Let U be the class of all recursive functions f that have string representations

$$0^m 1 \alpha \beta i j 0^\infty$$

where

(1) α and β are strings over $\{0, 1\}$, and
(2) $|\alpha| = |\beta|$, and
(3) if $\alpha = \beta$, or α follows β in the lexicographical ordering of strings over $\{0, 1\}$, then i is a program for the function, and
(4) if β follows α in the lexicographical ordering, then j is a program for the function, and
(5) $m = s_g(|\alpha| + |\beta|) - 1$.

Functions in U will be obtained by applying suitable recursion theorems. First, we consider some properties of g and h. We claim that there is a constant c such that, for all n, $g(s_g(n) + n) \leq c \cdot n$. Let n be sufficiently large so that for all $k \geq 0$, $n + k$ is in the range of g. Then $g(s_g(n)) = n$ (since n is in the range of g). We show that $g(s_g(n) + k) \leq n + k$ for all k. Suppose to the contrary that for some k, $g(s_g(n) + k) > n + k$. Since $g(s_g(n)) = n$, $g(s_g(n) + k) > n + k$, and since g is nondecreasing, the only arguments to g that could map to the k distinct values $n + 1, n + 2, \ldots, n + k$ (which are all in the range of g, by the choice of n) are arguments $s_g(n) + 1, s_g(n) + 2, \ldots, s_g(n) + (k - 1)$. A contradiction arises as there are only $k - 1$ arguments. We conclude that for sufficiently large n, $g(s_g(n) + k) \leq n + k$ for all k. It follows that for all sufficiently large n, $g(s_g(n) + n) \leq 2n$. Since g is defined only on integers, we can choose a suitable constant c such that for all $n \geq 1$ (in particular, even those not "sufficiently large"), $g(s_g(n) + n) \leq c \cdot n$.

Suppose by way of contradiction that $U \in EX(M) : h$. Recall that M's state information is contained in its long-term memory. For the constants c and k chosen above and sufficiently large n, M will have less than $n/2$ bits of memory to decide whether or not α preceeds β in the lexicographical ordering, for $n = |\alpha| + |\beta|$. For large enough n, c chosen so that $g(s_g(n) + n) \leq c \cdot n$, and $k > 2c$, $h(s_g(n) + n/2)$ is at most $1/k \cdot g(s_g(n) + n/2)$ (since h is $o(g)$) which is at most $1/k \cdot g(s_g(n) + n)$ (since g is nondecreasing). Finally, $1/k \cdot g(s_g(n) + n) < n/2$, by the choice of k. Hence, $h(s_g(n) + n/2) < n/2$. Let $m = s_g(n) - 1$. We will not choose α, α' and β, all of length $n/2$. Notice that after M has read the string $0^m 1 \alpha$ of length $s_g(n) + n/2$, it will have at most $h(s_g(n) + n/2) < n/2$ bits of long-term memory available. However, there are $2^{n/2}$ possible α's. Hence, we may choose α and α' such that M has the same long-term memory contents after seeing $0^m 1 \alpha$ as it does after seeing $0^m 1 \alpha'$. Let γ be the largest common prefix of α and α'. Suppose without loss of generality that $\gamma 1$ is a prefix of α and $\gamma 0$ is a prefix of α'. Choose $\beta = \alpha$. Note that α' preceeds β in the lexicographical ordering, while α does not.

By the mutual recursion theorem [Smullyan 1961], there are programs e_1 and e_2 such that program e_1 computes $0^m 1 \alpha \beta e_1 e_2 0^\infty$ and e_2 computes $0^m 1 \alpha' \beta e_1 e_2 0^\infty$. Notice that both φ_{e_1} and φ_{e_2} are in U. Furthermore, by the choice of α and α', the behavior of M on both functions is identical. Since the choice of n was arbitrary (as long as it is sufficiently large), for each of

infinitely many possible values of n, we have described a pair of functions, e_1^n and e_2^n such that if M uses at most $h(x_n)$ bits of long-term memory after reading the prefix of length $x_n = s_g(n) + n/2$, then one of the two functions cannot be inferred, and so M does not infer U. Any IIM M using f to bound long-term memory, must use more than $f(x_n) > h(x_n)$ long-term memory for each of the infinitely many choices of n (hence, x_n) which results in the infinitely many potentially look-alike pairs e_1^n and e_2^n in U. Hence, for infinitely many n, $f(n) > h(n)$.

The proof is completed by exhibiting an IIM M' such that $U \in EX(M') : g$. Notice that since for all n, $g(s_g(n)) \geq n$, by the fact that g is nondecreasing, $g(s_g(n) + k) \geq n \geq k$, for $k \leq n$. Consequently, for each $k \leq n$, $g(s_g(n) + k) \geq k$. Hence, M' can store the entire string $\alpha\beta$ in its long-term memory. Hence, M' will always be able to decide whether or not α preceeds β in the lexicographical ordering in the allotted space. Hence, $U \in EX(M') : g$. □

The next theorem reveals a gap phenomenon. There are classes of recursive functions that can either be learned in a constant amount of long-term space, or require at least a logarithmic amount of space.

THEOREM 5.4. *For any dense class U, for all IIM's M such that $U \in EX(M)$ either*:

(1) *there is a constant c such that $U \in EX(M) : c$, or,*
(2) *there is constant k such that for all n, there are infinitely many $f \in U$ with an initial segment, σ, of length n such that M, after reading σ as input, has used at least $(\log n) \cdot k$ long-term memory cells.*

PROOF. Suppose M is an IIM. Let s be the number of different symbols that may be written in a single cell of M's long-term memory. Suppose that there is no constant c satisfying (1) above. For any $d \in \mathbb{N}$, there is a fragment α such that M uses at least d cells of long-term memory to process. Pick a particular d and take a fragment α that forces M to use d cells of long-term memory. If α can be written as $\sigma\tau\rho$ where the contents of M's long-term memory is the same after reading $\sigma\tau$ as it was when it had finished reading σ, then let $\alpha' = \sigma\rho$. If α' can be similarly rewritten, form α'' by removing the segment that returns M to a prior long-term memory state. Eventually, a fragment β will be found such that M has a different long-term memory contents after reading each initial segment of β. Since there are $|\beta|$ initial segments,

$$|\beta| \leq s + s^2 + s^3 + \cdots + s^{d-1} = \frac{s^d - 1}{e - 1} \leq s^d.$$

Hence, $d \geq \log_s(|\beta|)$. The proof is completed by observing that each of $\beta, \beta 0, \beta 1, \beta 00, \beta 01, \ldots$ can be extended to a function in U. □

We note that Theorem 5.4 does not hold vacuously. To see this point, we need to construct two examples of dense classes. Let U be the class of *almost everywhere self-describing functions*. Formally, U consist of all the functions $f = \alpha e^\infty$ where α is some arbitrary initial segment and $\varphi_e = f$. Let $\sigma_0, \sigma_1, \ldots$ be an effective enumeration of all the finite initial segments. By the operator

recursion theorem [Case 1974], there is a recursively enumerable sequence of recursive functions, indexed by i, such that $\varphi_{h(i)} = \sigma_i h(i)^\infty$. Clearly, $\{\varphi_{h(i)} \mid i \in \mathbb{N}\} \subseteq U$ is dense. Furthermore, it can be learned by an IIM that simply copies the input to the output, an operation that involves no long-term memory.

The second example of a dense class will be one that requires at least logarithmic long-term memory. This class is like the almost everywhere self describing functions, except that the self description is done in the unary alphabet. Formally, U' consists of all the functions $f = \alpha(01^e)^\infty$ where α is some arbitrary initial segment and $\varphi_e = f$. Another operator recursion theorem argument [Case 1974] constructs functions $\varphi_{h(i)} = \sigma_i(01^{h(i)})^\infty$ all of which are in U'. Clearly, U' is dense. Furthermore, to learn U', an IIM must count the lengths of the sequences of 1's. Logarithmic long-term space is sufficient for this task. To show that logarithmic long-term space is *required*, we need only show that U' cannot be learned in constant long-term space and appeal to Theorem 5.4. To do this, we use an argument similar to one used in the proof of Theorem 5.3. Suppose that M is an IIM with $U' \in EX(M) : c$. First we find two different strings α_0 and α_1, of the same length, such that M is the same memory configuration after reading α_0 as it does after reading α_1. Since M's memory is assumed to be of constant size, two such strings will exist. By two applications of the recursion theorem [Kleene 1938], there are programs e_0 and e_1 such that $\varphi_{e_0} = \alpha_0(01^{e_0})^\infty$ and $\varphi_{e_1} = \alpha_1(01^{e_1})^\infty$. Clearly, both φ_{e_0} and φ_{e_1} are in U'. M will exhibit the same limiting behavior on both functions, contradicting the assumption that $U' \in EX(M) : c$.

As is evident from the proof of the Theorem 5.1, the amount of long-term memory used by an IIM to infer some function may vary depending on the order of presentation of the graph of the function. This variation may swing from one extreme to the other, as evidenced by the following theorem. For the following theorem, the IIMs are assumed to input ordered pairs, not just elements of the range.

THEOREM 5.5. *There is a class U of total recursive functions and a particular enumeration of functions in U such that there is an IIM M witnessing $U \in EX(M) : c$ with respect to this order. Furthermore, with respect to the natural, increasing domain order, $U \notin EX(M') : g$ for any M' and any sublinear recursive function g.*

PROOF. Let U be the set of recursive functions f such that if n is the smallest number such that $f(2n + 1) = 1$, then either $n = 0$ and $f(x) = 0$ for all $x > 1$, or else $n \neq 0$ and $\varphi_{f(2n)} = f$.

The unusual ordering that we will use to infer U with a constant amount of memory is the so called *reverse pair ordering*: $f(1), f(0), f(3), f(2), f(5), f(4), \ldots$. This ordering will be compared with the natural, increasing domain order: $f(0), f(1), f(2), f(3), \ldots$. Next, we describe the behavior of the IIM M that only uses long term memory (3 bits) to remember which state it is in and infers all the functions $f \in U$, provided the input is received in the reverse pair ordering.

Operation of M (starting in state 1):

(1) Read an input $f(y)$ into the short-term memory. If this is the reverse pair ordering, this value will be $f(1)$. If the value in short-term memory is "1", then go to state 2, else go to state 3.

(2) Read an input $f(y)$ into the short-term memory. If this is the reverse pair ordering, this value will be $f(0)$. Let z be the value currently in short-term memory. Output a program for the following function ψ:

$$\psi(x) = \begin{cases} z & \text{if } x = 0, \\ 1 & \text{if } x = 1, \\ 0 & \text{otherwise.} \end{cases}$$

After outputting this program, stop. (State 2 is a halting state.)

(3) Read an input $f(y)$ into the short-term memory. If this is the reverse pair ordering, then y will be an even number. Go to state 4.

(4) Read an input $f(y)$ into the short-term memory. If this is the reverse pair ordering, then y will be an odd number. If the value in short-term memory is "1" then go to state 5, else go to state 3.

(5) Read an input $f(y)$ into the short-term memory. Output this same value and stop.

Suppose that $f \in U$ is given to M in the reverse pair ordering. Then M will first read the value of $f(1)$ (state 1). If this value is a "1", then M will read the value of $f(0)$, output the correct program and halt in state 2. If the value of $f(1) \neq 1$, then M reads data (states 3 and 4) until it finds $f(y) = 1$ for some odd value of y in state 4. Then M goes to state 5, and reads the correct program in as the next input.

The proof is completed by showing that no IIM can learn U with sublinear memory with respect to the natural, increasing domain order. Let M' be an IIM and suppose by way of contradiction that $U \subseteq EX(M') : g$, with respect to the increasing domain order, for g a sublinear recursive function. By the operator recursion theorem [Case 1974], there is a monotone increasing recursive function h such that, for any i,

$$\varphi_{h(i)}(x) = \begin{cases} h(i) & \text{if } x = 0, \\ 1 & \text{if } x = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, each $\varphi_{h(i)}$ is in U. Since h is monotone increasing, for any i and j, $h(i) \neq h(j)$ and $\varphi_{h(i)} \neq \varphi_{h(j)}$. Then, for any n, M' must remember $\varphi_{h(0)}(0), \varphi_{h(1)}(0), \ldots, \varphi_{h(n)}(0)$, differently. Consider $n = 2^b$, for some b. So there are 2^b different values to be saved in order to correctly learn $\varphi_{h(0)}, \ldots, \varphi_{h(n)}$. This will require at least b bits. Hence, M' requires at least linear long-term memory. □

6. *Probabilistic Limited Memory Machines*

Probabilistic inductive inference machines were introduced in Pitt [1989] and studied further in Pitt and Smith [1988]. A *probabilistic* inductive inference machine is an IIM that makes use of a fair coin in its deliberations. We say that $f \in EX(M)\langle p \rangle$ if M learns f with probability p, $0 \leq p \leq 1$. The collection $EX\langle p \rangle$ is defined to be $\{U \mid \exists M(U) \subseteq EX(M)\langle p \rangle\}$. Pitt showed that for $p > \frac{1}{2}$, $EX\langle p \rangle = EX$ [Pitt 1989]. Limiting the memory available to a probabilistic IIM, according to the conventions of this paper, gives rise to the class $EX : c\langle p \rangle$.

In this section, we define a class of recursive functions, called $\bar{U}$, and prove three theorems about it. Firstly, we show that $\bar{U}$ can be probabilistically learned (with probability 1) by an algorithm that uses only a constant amount of long-term memory. Then we show that both the upper and lower bounds for learning $\bar{U}$ deterministically are logarithmic. We proceed with the definition of $\bar{U}$.

Every function in $\bar{U}$ will take on only four values, 0, 1 and two self-referential indices. Members of $\bar{U}$ will be constructed via suitable recursion theorems. Every function $f \in \bar{U}$ will have several (perhaps infinitely many) blocks of 0's. Let $\tau_1, \tau_2, \ldots$ denote the length of the first block of 0's, the second block, etc. Similarly, $\sigma_1, \sigma_2, \ldots$ denotes the lengths of the blocks of 1's, in their order of appearance in the range. For a function f to be in $\bar{U}$, one of the following two conditions must be met:

$$\sum_{n=1}^{\infty} \frac{1}{2^{\tau_n}} \text{ converges and } \sum_{n=1}^{\infty} \frac{1}{2^{\sigma_n}} \text{ diverges, or} \tag{1}$$

$$\sum_{n=1}^{\infty} \frac{1}{2^{\tau_n}} \text{ diverges and } \sum_{n=1}^{\infty} \frac{1}{2^{\sigma_n}} \text{ converges.} \tag{2}$$

Furthermore, if case (1) occurs, $f \in \bar{U}$ iff the sequence of values $f(x_1)$, $f(x_2), \ldots$, for positions x_i immediately following a block of 0's or 1's, converges to a program for f. Similarly, for (2) to qualify a function f for membership in $\bar{U}$, the sequence of values $f(y_1), f(y_2), \ldots$ converges to a program for f, where $y_i = x_i + 1$, for example the y_i's are points immediately following a point that immediately follows a block of 0's or 1's.

THEOREM 6.1. $\bar{U} \in EX : c\langle 1 \rangle$.

PROOF. The proof proceeds by constructing two probabilistic ω-automata [Ablaev and Freivalds 1986; Taimina and Freivalds 1966]. These ω-automata will process the string of values representing the range of functions from $\bar{U}$. Consequently, they will only have to recognize symbols as being either 0 or 1 or other. The state transition graph of automata A_1 and A_2 are given in figures 1 and 2 respectively. The arc from q_1 to q_2 labeled 1, $\frac{1}{2}$ indicates, that when in state q_1, if the automaton sees a 1, it enters state q_2 with probability $\frac{1}{2}$.

Let some function f from $\bar{U}$ be given. Consider what happens when the string representation of f is given as input to A_1 and A_2. What will turn out to be important is the number of times A_1 enters state q_3 and how often A_2 enters state q_3'. Our discussion will be in terms of A_1, q_3 and 1's. The same dialogue will hold for A_2 with q_i replaced by q_i' and all references to 1 replaced by 0.

State q_3 can only be entered from state q_1. State q_1 is accessible only from states q_3 and q_4. Observation of a 1 is required for a change of state to q_1. Once in state q_1, observing more 1's can keep A_1 in the same state. When A_1 is any state other than q_2, observation of a 1 will move the automaton to state q_1 or state q_2 with equal probability. From state q_2, observing a 1 keeps the automaton in state q_2, thus if A_1 observes n consecutive 1's, then the probability that it will be in state q_1 after seeing them all is at most 2^{-n}.

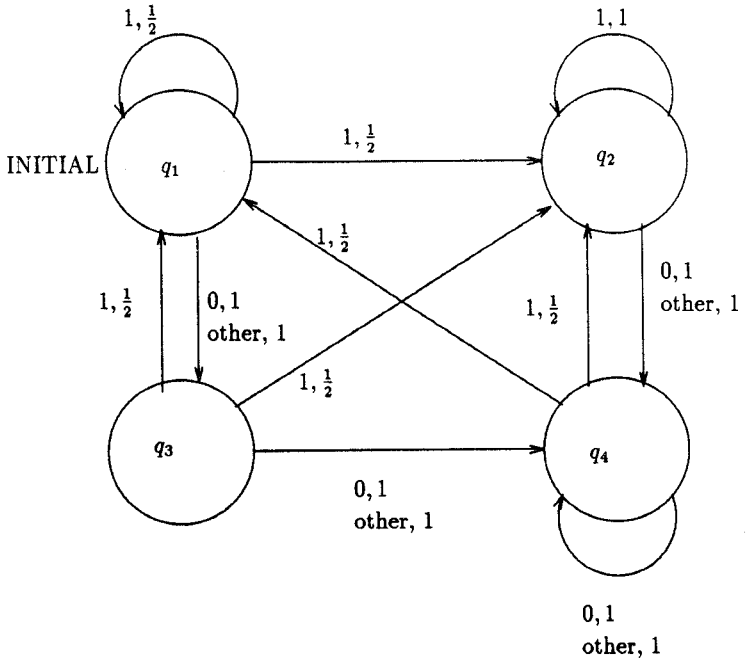

FIG. 1. Automaton A_1.

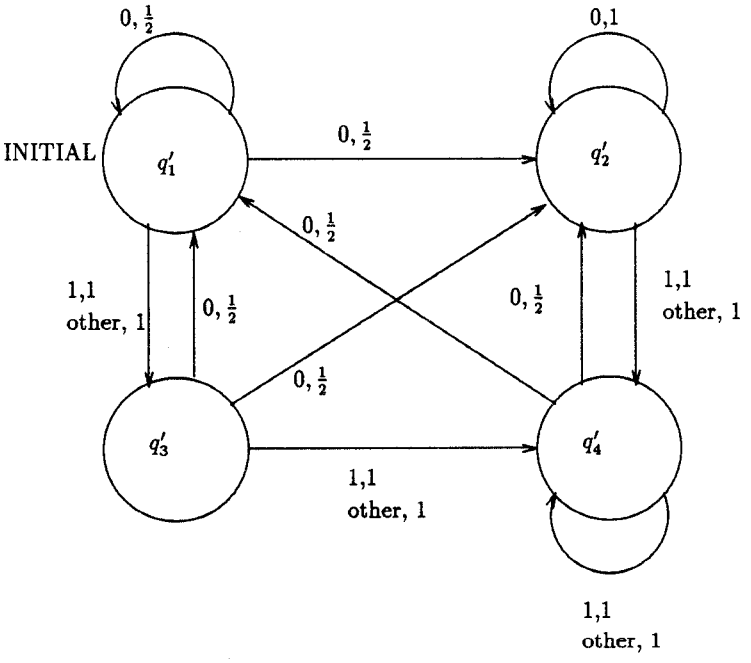

FIG. 2. Automaton A_2.

Recall that σ_i denotes the length of the ith block of 1's in the string representation of f. By the above discussion, if

$$\sum_{n=1}^{\infty} \frac{1}{2^{\sigma_n}} \tag{3}$$

diverges, then by the Borel–Cantelli lemma [Feller 1968], state q_1, hence state q_3 will be entered just at the end of reading an entire block infinitely often with probability 1, and finitely often with probability 0. On the other hand, if (3) converges, then, with probability 1, state q_3 will be entered only finitely often (and infinitely often with probability 0). This is because q_3 can only be entered by reading a 0 or "other" at the end of a block of 1's, and this happens *only* if at the end of the block, A_1 was in state q_1.

Similarly, in A_2, state q_3' will, with probability 1, be entered infinitely often when

$$\sum_{n=1}^{\infty} \frac{1}{2^{\tau_n}} \tag{4}$$

diverges (finitely often with probability 0). Since $\overline{U}$ was defined to include only functions f such that (3) converges iff (4) diverges, the following statement holds with probability 1 for any member of $\overline{U}$:

> either state q_3 is entered infinitely often and q_3' only finitely often or state q_3' is entered infinitely often and q_3 only finitely often.

A probabilistic IIM, M, that infers all the functions in $\overline{U}$ simulates the behavior of A_1 and A_2, tossing a fair coin every time a probabilistic state transition is made. Suppose that $f \in \overline{U}$. If observing the value $f(x)$ causes A_2 to enter state q_3', then M outputs the value $f(x)$ when it is received as input. If observing the value $f(x+1)$ causes A_1 to enter state q_3, then M outputs $f(x+2)$ when it is received as input. Under no other circumstances will the IIM produce an output.

With probability 1, M converges to a program that computes f. The memory used by M is bounded by a constant. All that is needed is to remember the transitions of A_1 and A_2, their current states, two bits to remember if q_3 or q_3' was entered, and one final bit to be able to count 2 more inputs before transferring an input to the output tape. □

THEOREM 6.2. *Suppose that $h = o(\log n)$ for all IIMs M, $\overline{U} \notin EX(M) : h$.*

PROOF. Suppose $h = o(\log n)$. Suppose by way of contradiction that M is an IIM such that $\overline{U} \subseteq EX(M) : h$. Define $a_n = \log n$. Let $LCM(n)$ denote the least common multiple of $\{1, \ldots, n\}$. We would like to define $b_n \geq a_n + LCM(n)$. To do so, we must develop an upper bound for $LCM(n)$. From elementary number theory (e.g., see Griffin [1954]), the number of primes less than or equal to n is $O(n/\ln n)$. The largest possible factor of any prime in $LCM(n)$ is n. Consequently, an upper bound for $LCM(n)$ is

$$O(n^{O(n/\log n)}).$$

Thus, we can choose $b_n = c \log n^{c \log n / \ln \log n}$ for some constant c. Consequently,

$$\sum_{n=1}^{\infty} \frac{1}{2^{a_n}}$$

diverges and

$$\sum_{n=1}^{\infty} \frac{1}{2^{b_n}}$$

converges.

The sequences of a_i's and b_i's will be used to determine the sizes of the blocks of consecutive 0's and 1's in the functions that we will construct below. Before defining the function from $\overline{U}$ that M will fail to identify, we must first describe a transformation on programs. Recall that functions in $\overline{U}$ have string representations that look like:

$$0^{a_1}xy1^{b_1}xy0^{a_2}xy1^{b_2}xy \cdots .$$

If φ_i has a string representation that conforms to the above schema, then $\varphi_{g(i)}$ will interchange the roles of a_k and b_k for $k > c$ resulting in the following appearance:

$$0^{a_1}xy1^{b_1}xy \cdots xy0^{a_c}xy1^{b_c}xy0^{b_{c+1}}xy1^{a_{c+1}}xy0^{b_{c+2}}xy1^{a_{c+2}}xy \cdots .$$

The transformation g is specified by the construction of $\varphi_{g(i)}$, uniformly in i in effective stages of finite extension below. Consequently, g will be a total recursive function, even if some of the programs in its range compute functions with finite (or empty) domains. $\varphi^s_{g(i)}$ denotes the finite amount of $\varphi_{g(i)}$ determine prior to stage s. $\varphi^0_{g(i)} = \varnothing$. x^s denotes the least number not in the domain of $\varphi^s_{g(i)}$. Consequently, $x^0 = 0$.

Begin stage 0. Look for the least z such that $\varphi_i(z) = 0$ and there are $2 \cdot c$ numbers $y < z$ such that $\varphi_i(y)$ is defined to some number other than 0 or 1. Set $\varphi^1_{g(i)} = \{(x, \varphi_i(x) \mid x < z\}$ and go to stage 1.
End stage 0.

Begin stage $s > 0$. Look for the least $z \geq x^s$ such that $\varphi_i(z)$ is defined and $\varphi_i(z) \neq 0$. If $\varphi_i(z)$ is undefined for some value, then $\varphi_{g(i)}$ has finite domain. Look for the least $w > z + 1$ such that $\varphi_i(w)$ is defined and

$\varphi_i(w) \neq 1$. Again, if φ_i is undefined for some value involved in the search, then $\varphi_{g(i)}$ has finite domain. Define:

$$\varphi^{s+1}_{g(i)}(x) = \begin{cases} \varphi^s_{g(i)}(x) & \text{if } x < x^s \\ 0 & \text{if } x^s \leq x < x^s + w - (z+2) \\ \varphi_i(z) & \text{if } x = x^s + w - (z+2) \\ \varphi_i(z+1) & \text{if } x = x^s + w - (z+2) + 1 \\ 1 & \text{if } x^s + w - (z+2) + 1 < x < w \\ \varphi_i(w) & \text{if } x = w \\ \varphi_i(w+1) & \text{if } x = w + 1 \end{cases}$$

Go the stage $s + 1$.
End stage s.

We are now ready to define the function $f \in \overline{U}$ that cannot be identified by M. By implicit use of the recursion theorem, define a function φ_e with string representation:

$$0^{a_1}g(e)e1^{b_1}g(e)e0^{a_2}g(e)e1^{b_2}g(e)e \cdots .$$

Clearly, φ_e is a total function. By the choice of the a_i's and b_i's, $\varphi_e \in \overline{U}$. Let n_i denote the length of the string

$$0^{a_1}g(e)e1^{b_1}g(e)e \cdots 0^{a_i}g(e)e1^{b_i}g(e)e.$$

Let $f = \varphi_e$. To complete the proof, we define a recursive function f' that is different from f but that M cannot distinguish from f. The argument is similar to a pumping lemma argument. Notice that $a_i = \Omega(\log n_i)$. Each block of 0's (length a_i) and block of 1's (length b_i) is longer than $h(n_i)$ in length, for sufficiently large i. (The convergence or divergence of the series is not dependent on the first few values n_i.) Choose an i such that $h(x) \leq c \cdot \log(x)$ for all $x \geq n_i$. So, for large enough i there is a block of $d_i \leq h(n_i)$ 0's such that M's memory (and internal state) are in the same configuration when those 0's are just about to be read, and just after they have all been read. Similarly, there is a block of $j_i \leq h(n_i)$ 1's. Redrawing the string representation of f, given the above observation yields:

$$\cdots \underbrace{000 \cdots 0000^{d_i}00 \cdots 000}_{a_i} g(e)e \underbrace{111 \cdots 1111^{j_i}111 \cdots 1111}_{b_i} \cdots.$$

Since $b_i - a_i = LCM(i)$, both d_i and j_i divide $b_i - a_i$, for large enough i. Consequently, for such i's, it is possible to expand the ith block of 0's by a multiple of d_i to make the block have exactly b_i 0's. Similarly, it is possible to remove a multiple of j_i 1's from the ith block of 1's, leaving the block with exactly a_i 1's. Performing this transformation on all but finitely many blocks of 0's and 1's results in a function that we will call f'. Notice that the first value following each block of 0's or 1's ($g(e)$) is an index for f'; hence, $f' \in \overline{U}$. M, on input from f' will be in the same state when it leaves the nth block of 0's (or 1's) as it will be when using f as input. Consequently, M will produce the same outputs on input from f and f'. Hence, M cannot infer both f and f', a contradiction to the assumption that $\overline{U} \subseteq EX(M) : h$. □

THEOREM 6.3. *Let* $g = \lambda x[3 \cdot \log x]$. *Then* $\overline{U} \in EX : g$.

PROOF. Let $\overline{U}$ and g be as in the hypothesis. The key to inferring some $f \in \overline{U}$ is to decide which of the two series:

$$\sum_{n=1}^{\infty} \frac{1}{2^{\tau_n}} \tag{5}$$

$$\sum_{n=1}^{\infty} \frac{1}{2^{\sigma_n}} \tag{6}$$

converges.

We will estimate the values is the series (5) and (6). This estimate will be kept in long-term memory. The series with the smaller estimate will be assumed to be the one that is converging. Suppose that we have an estimate of the sum of the first $t - 1$ values of both series. The value of $1/2^{\tau_t}$ can be obtained by reading τ_t zeros into short-term memory. If $\tau_t > \log t + 2 \log\log t$ (e.g., $1/2^{\tau_t} < 2^{-\log t - 2\log\log t}$), then this value is ignored, hence at most a logarithmic amount of long-term memory will be used for this purpose. Otherwise, the value of $1/2^{\tau_t}$ is added to the estimate of (5) in long-term memory. The value of (6) is updated similarly. We add to the sums numbers of

lengths at most $2 \log t \geq \log t + 2 \log \log t$. For the first t numbers we are adding numbers of lengths $2 \log 1, 2 \log 2, 2 \log 3, \ldots, 2 \log t$. Since there are t such numbers, each of value at most t^2, the sum is at most t^3, so the space needed for the sum is at most $3 \log t$. The above argument is relevant for both series (5) and (6).

The proof is completed by showing that the terms that are ignored do not have any effect on the convergence or divergence of the series. Since the series

$$\sum_{n=1}^{\infty} \frac{1}{n(\log n)^2}$$

converges, there is no need to consider terms smaller than $1/t(\log t)^2$. This happens precisely when $\tau_t > \log t + 2 \log \log t$. □

7. *Conclusions*

A complexity theoretic model of learning in the limit without complete information was presented. This model was related to previously studied restrictions on the traditional model of learning in the limit. It was shown that linear space is enough long-term memory to complete any learning task. With logarithmic long-term space, it is possible to maintain a counter. Without this much space, what can be accomplished seems to be done via finite automata, requiring only a constant amount of space. Several lower bound results were given. The proof of these results used an unusual combination of pumping arguments with mutual recursion theorems. A space hierarchy result was also shown. Probabilistic memory limited learning was also examined.

There are many interesting questions concerning the trade-offs between long- and short-term memory. We have been able to obtain only some preliminary results concerning the relationships between long- and short-term memory sizes for certain problems [Freivalds et al. 1993b].

The model used in all these results required that the short-term memory be destroyed each time the learning machine starts to read another datum. An alternative model would be for the data in short-term memory to be destroyed every time a new bit is read. Some of our results may not hold in this alternative model.

Most of our results, with the exception of Theorem 5.5, considered giving the inputs to an IIM in the natural, increasing domain order. Although this is a traditional starting point, it would be desirable to consider the worst case over all possible orders. Certainly, our lower-bound results, as well as some results concerning constant long-term memory, would hold in this model. The other theorems do not hold in any obvious way. A further investigation of worst-case memory utilization would be very interesting.

ACKNOWLEDGMENTS. Some results in this paper were obtained while the authors attended ICALP'91 in Madrid. We gratefully acknowledge The Universities of Latvia and Maryland, the National Science Foundation and the organizers of ICALP'91 for making it possible for the authors to convene in a setting conducive to obtaining the results described in this paper. Our colleague, Bill Gasarch, made valuable comments on an early draft of this paper. Sally Goldman served as a guide to the PAC learning literature. Haym Hirsh pointed out the memory utilization scheme of Soar to us. Simon Kasif, Richard

Lewis, Don Perlis, Paul Rosenbloom, and Steven Salzberg also made comments on earlier drafts. The very thorough referees made several valuable suggestions, greatly improving this paper.

REFERENCES

ABLAEV, F. M., AND FREIVALDS, R. 1986. Why sometimes probabilistic algorithms can be more effective. In *Lecture Notes in Computer Science*, vol. 233. Springer Verlag, New York, pp. 1–14.

AMEUR, F., FISCHER, P., HÖFFGEN, K., AND AUF DER HEIDE, F. 1993. Trial and error: A new approach to space-bounded learning. In *Proceedings of the 1st European Conference on Computational Learning Theory*. Oxford University Press, Oxford, England, pp. 133–144.

ANDERSON, J. R. 1983. *The Architecture of Cognition*. Harvard University Press, Cambridge, Mass.

ANGLUIN, D., AND SMITH, C. H. 1983. Inductive inference: Theory and methods. *Comput. Surv. 15*, 237–269.

ANGLUIN, D., AND SMITH, C. H. 1987. Inductive inference. In *Encyclopedia of Artificial Intelligence*, S. Shapiro, ed. Wiley, New York, pp. 409–418.

BAR-HILLEL, V., PERLES, M., AND SHAMIR, E. 1951. On formal properties of simple phrase structured grammars. *Z. Phon. Sprach., Kommun. 14*, 143–172.

BARZDINS, J. 1974. Two theorems on the limiting synthesis of functions. In *Theory of Algorithms and Programs*, vol. 1. J. Barzdins, ed. Latvian State University, Riga, U.S.S.R., pp. 82–88.

BLUM, L., AND BLUM, M. 1975. Toward a mathematical theory of inductive inference. *Inf. Cont. 28*, 125–155.

BOUCHERON, S., AND SALLANTIN, J. 1988. Some remarks about space-complexity of learning, and circuit complexity of recognizing. In *Proceedings of the 1988 Workshop on Computational Learning Theory*. D. Haussler and L. Pitt, eds. Morgan-Kaufmann, San Mateo, CA, pp. 125–138.

BRAINE, M. D. S. 1971. On two types of models of the internalization of grammars. In *The Ontogenesis of Grammar*, D. I. Slobin, ed. Academic Press, Orlando, Fla., pp. 153–186.

CASE, J. 1974. Periodicity in generations of automata. *Math. Syst. Theory 8*, 15–32.

CASE, J., JAIN, S., AND SHARMA, A. 1994. Convergence to nearly minimal size grammars by vacillating learning machines. *J. Comput. Syst. Sci. 49*, 2, 189–207.

CASE, J., AND NGOMANGUELLE, S. Refinements of inductive inference by popperian machines. *Kybernetika*, to appear.

CASE, J., AND SMITH, C. 1983. Comparison of identification criteria for machine inductive inference. *Theoret. Comput. Sci. 25*, 2, 193–220.

CRICK, F., AND MITCHISON, G. 1983. The function of dream sleep. *Nature 304*, 14, 111–114.

FELLER, W. 1968. *An Introduction to Probability Theory and Its Applications*, vol. 1. Wiley, New York.

FLOYD, S. 1989. Space-bounded learning and the Vapnik–Chervonenkis dimension. In *Proceedings of the 1989 Workshop on Computational Learning Theory*, R. Rivest, D. Haussler, and M. Warmuth, eds. Morgan-Kaufmann, San Mateo, Calif., pp. 349–364.

FREIVALDS, R. 1990. Inductive inference of minimal size programs. In *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*, M. Fulk and J. Case, eds. Morgan Kaufmann, San Mateo, Calif., pp. 1–20.

FREIVALDS, R., KINBER, E., AND SMITH, C. 1993a. Learning with a limited memory. In *Notes of the AAAI Spring Symposium on Training Issues in Incremental Learning*. Stanford Univ., Stanford, Calif., pp. 78–87.

FREIVALDS, R., KINBER, E., AND SMITH, C. 1993b. On the impact of forgetting on learning machines. In *Proceedings of the 6th Annual ACM Conference Computational Learning Theory* (Santa Cruz, Calif., July 26–28). ACM, New York, pp. 165–174.

FREIVALDS, R., KINBER, E., AND SMITH, C. 1993c. Probabilistic versus deterministic memory-limited learning. In *Record of the Workshop on Algorithmic Learning for Knowledge Processing*.

FREIVALDS, R., KINBER, E., AND SMITH, C. 1994. Quantifying the amount of relevent information. In *Notes of the AAA1 Spring Symposium on Reference*, pp. 76–79.

FREIVALDS, R., AND SMITH, C. 1992. Memory limited inductive inference machines. In *Lecture Notes in Computer Science*, vol. 621. Springer-Verlag, New York, pp. 19–29.

FREIVALDS, R., AND SMITH, C. 1993. On the power of procrastination for machine learning. *Inf. Comput. 107*, 237–271.

FULK, M., AND CASE, J., EDS. 1990. *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*. Morgan-Kaufmann, San Mateo, Calif.

GOLD, E. M. 1967. Language identification in the limit. *Inf. Cont. 10*, 447–474.

GRIFFIN, H. 1954. *Elementary Theory of Numbers*. McGraw-Hill, New York.

HAUSSLER, D. 1985. Space efficient learning algorithms. Tech. Rep. UCSC-CLR-88-2. Univ. California at Santa Cruz, Santa Cruz, Calif.

HAUSSLER, D., ED. 1992. *Proceedings of the 5th Annual Workshop on Computational Learning Theory* (Pittsburgh, Pa., July 27–29). ACM, New York.

HAUSSLER, D., AND PITT, L., EDS. 1988. *Proceedings of the 1988 Workshop on Computational Learning Theory*. Morgan-Kaufmann, San Mateo, Calif.

HEATH, D., KASIF, S., KOSARAJU, R., SALZBERG, S., AND SULLIVAN, G. 1991. Learning nested concept classes with limited storage. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence* (Sydney, Australia). Morgan-Kaufmann, San Mateo, Calif., pp. 777–782.

HELMBOLD, D., SLOAN, R., AND WARMUTH, M. 1989. Learning nested differences of intersection-closed concept classes. In *Proceedings of the 1989 Workshop on Computational Learning Theory*, R. Rivest, D. Haussler, and M. Warmuth, eds. Morgan-Kaufmann, San Mateo, Calif., pp. 41–56.

HOPFIELD, J. J., FEINSTEIN, D. I., AND PALMER, R. G. 1983. 'Unlearning' has a stabilizing effect in collective memories. *Nature 304*, 14, 158–159.

JANTKE, K. P., AND BEICK, H. R. 1981. Combining postulates of naturalness in inductive inference. *Electron. Inf. Kyber. 17*, 465–484.

KLEENE, S. 1938. On notation for ordinal numbers. *J. Symb. Logic 3*, 150–155.

LEVINE, D. 1991. *Introduction to Neural and Cognitive Modeling*. Lawrence Earlbaum Associates, Hillsdale, N.J.

LEWIS, H., AND PAPADIMITRIOU, C. 1981. *Elements of the Theory of Computation*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

LIN, J. H., AND VITTER, J. S. 1994. A theory for memory-based learning. *Mach. Learn. 17*, 2, 143–168.

MACHTEY, M., AND YOUNG, P. 1978. *An Introduction to the General Theory of Algorithms*. North-Holland, New York.

MICHALSKI, R., CARBONELL, J., AND MITCHELL, T. 1983. *Machine Learning*. Tioga Publishing Co., Palo Alto, Calif.

MILLER, G. 1956. The magical number seven plus or minus two. *Psych. Rev. 63*, 81–97.

MIYAHARA, T. 1987. Inductive inference by iteratively working and consistent strategies with anomalies. *Bull. Inf. Cybern. 22*, 171–177.

MIYAHARA, T. 1989. A note on iteratively working strategies in inductive inference. In *Proceedings of the Fujitsu IIAS-SIS Workshop on Computational Learning Theory* (Numazu, Japan).

OSHERSON, D., STOB, M., AND WEINSTEIN, S. 1986. *Systems that Learn*. MIT Press, Cambridge, Mass.

PITT, L. 1989. Probabilistic inductive inference. *J. ACM 36*, 2 (Apr.), 383–433.

PITT, L., AND SMITH, C. 1988. Probability and plurality for aggregations of learning machines. *Inf. Comput. 77*, 77–92.

POPPER, K. 1968. *The Logic of Scientific Discovery*. Harper Torch Books, New York.

RIVEST, R., HAUSSLER, D., AND WARMUTH, M., EDS. 1989. *Proceedings of the 2nd Annual Workshop on Computational Learning Theory* (Palo Alto, Calif.). Morgan-Kaufmann, San Mateo, Calif.

ROGERS, JR., H. 1967. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York.

ROSENBLOOM, P., LAIRD, J., NEWELL, A., AND MCCARL, R. 1991. A preliminary analysis of the Soar architecture as a basis for general intelligence. *Artif. Int. 47*, 289–325.

SCHAPIRE, R. 1990. The strength of weak learnability. *Mach. Learn. 5*, 2, 197–227.

SERVAN-SCHREIBER, E. 1991. *The Competitive Chunking Theory: Models of Preception, Learning, and Memory*. Ph.D. dissertation. Department of Psychology, Carnegie Mellon Univ., Pittsburgh, Pa.

SHAPIRO, S. 1987. *Encyclopedia of Artificial Intelligence*. Wiley, New York.

SIEGELMANN, H. T., AND SONTAG, E. D. 1992. On the computational power of neural nets. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory* (Pittsburgh, Pa., July 27–29). ACM, New York, pp. 440–449.

SMULLYAN, R. 1961. *Theory of Formal Systems, Annals of Mathematical Studies*, vol. 47. Princeton University Press, Princeton, N.J.

TAIMINA, D. YA., AND FREIVALDS, R. 1966. On complexity of probabilistic finite automata recognizing superlanguages. In *Methods of Logic in Construction of Effective Algorithms*. Kalinin State Univ., Tver, pp. 92–96.

VALIANT, L. G. 1984. A theory of the learnable. *Commun. ACM 27*, 11 (Nov.), 1134–1142.

WARMUTH, M., AND VALIANT, L., EDS. 1991. *Proceedings of the 1991 Workshop on Computational Learning Theory* (Palo Alto, Calif.). Morgan-Kaufmann, San Mateo, Calif.

WEXLER, K., AND CULICOVER, P. W. 1980. *Formal Principles of Language Acquisition*. The MIT Press, Cambridge, Mass.

WEIHAGEN, R. 1976. Limes-erkennung rekursiver funktionen durch spezielle strategien. *Elek. Inf. Kyber. 12*, 93–99.

RECEIVED JULY 1993; REVISED JUNE 1995; ACCEPTED JULY 1995