Quantifying the Amount of Relevant Information *

Rūsiņš Freivalds
Inst of Math and Comp Sci
University of Latvia
Raiņa bulvāris 29
LV-1459, Riga, Latvia

Efim Kinber†
Dept of Comp and Info Sci
University of Delaware
Newark, DE 19716

Carl H. Smith‡
Dept of Comp Sci
University of Maryland
College Park, MD 20912 USA

The expense of maintaining all observed data during a learning episode has inspired many researchers to consider various strategies to reduce the amount of data retained. Our work introduces the first true complexity theoretic analysis of space utilization by learning algorithms that repeatedly input examples and produce new hypotheses. In our new model, memory is measured in bits as a function of the size of the input. There is a hierarchy of learnablity 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 is necessary consider two types of memory: long and short term. Any trade offs between the two types of memory are shown to be limited.

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 and Pitt, 1988; Rivest *et al.*, 1989; Fulk and Case, 1990; Warmuth and Valiant, 1991; ACM, 1992; ACM, 1993; ACM, 1994]. 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 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. Several authors have argued that such studies are general enough to include a wide array of learning situations [Angluin and Smith, 1983; Angluin and Smith, 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. For the purposes 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

*This work was facilitated by an international agreement under NSF Grant 9119540.

†On leave from Institute of Mathematics and Computer Science, University of Latvia, Riga, Latvia

‡Supported in part by NSF Grants 9020079 and 9301339.

on the model of Gold [Gold, 1967] that was cast recursion theoretically in [Blum and Blum, 1975].

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 inputs the range of a recursive function, an ordered pair at a time, and, while doing so, outputs computer programs. Since we will only discuss the inference of (total) recursive functions, we may assume, without loss of generality, that the input is received by an IIM in its natural domain increasing order, $f(0)$, $f(1)$, $\cdots$. An IIM, on input from a function f will output a potentially infinite sequence of programs p_0, p_1, $\cdots$. 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 $f$if, when M is given the range of f as input, it converges to a program p that computes f. If an 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. 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.

Several authors from the fields of cognative science, linguisitics, connectionism, PAC learnablility and inductive inference considered memory limited learning. Most of these prior models consider only the accounting of the number of data items remembered, independent of their size. Those models that do consider, in some way, the size of the data remebered, so not take into account the memory requirements of remembering other pertinent information such as prior hypotheses and state information for the algorithm under investigation. Also, in many of the previous studies, the complexity function was curiously not a function of input size.

We now describe the model investigated in this paper. To insure 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, as opposed to integers. Each data entry will appear as a bit string with a designated delimeter 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. Including the state information in the memory that we are accounting for is a major departure from previous work. None of the previous models of memory limited learning even considers the state information of the learning algorithm itself. If the number of states of a learning algorithm was unbounded, then the possiblity of encoding information into the state space arises. For example, given a set of states, s_0, $\cdots$, s_n, of some learning algorithm, consider another learning algorithm that has states s_i^j for $1 \leq i \leq n$ and $0 \leq j \leq 1,000,000,000$. One interpretation is that state s_i^j means that the original algorithm would be in state s_i and it would have seen data encoded by j. If the states are not assumed to consume any space, then an arbitrary amount of information can be encoded into the state space without impacting the memory utilization. As an added benefit to including the state information in our

long term memory, we can use pumping techniques from formal language theory [Lewis and Papadimitriou, 1981] in our proofs.

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 insure an accurate accounting of the real long term memory utilization of a learning process. It might be, as indicated by our results in the section on trade offs between long and short term memory, 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 need by a learning algorithm. The introduction of the the short term memory enabled us to get sharper lower bound results. Most of our results, however, are stated in terms of long term memory requirements, with the short term memory appearing only in the proof.

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-Schreiber, 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] to 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.

In the remaining space, we state some of our results [Freivalds and Smith, 1992; Freivalds *et al.*, 1993]. It turns out that linear space, with very small constants, is sufficent to be able to learn anything that can be learned. Furthermore, sometimes, linear long term space is required. we also have several examples of classes requiring logarithmic lower bounds on long term memory. One of our more complicated examples of a class requireing logarithmic space has the property that it can be learned by a *probabilistic* learning algorithm, in constant space, with probability 1.

There is a hierarchy, based on larger and larger space allotments. 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. 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 than can be learned in g space, but not in h space. For the contrived example g space learnable set of functions that we use in the proof, it can be shown that to learn the set, the bound of h space is exceeding infinitely often. The proof uses a novel combination of pumping arguments from formal language theory [Lewis and Papadimitriou, 1981] and mututal recursion arguments from recursion theory [Soare, 1987].

In contrast to the hierarchy result, we also have discovered a gap phenomeneon for certain types of classes. A class of functions is *dense* if every finite data set can be extended to form some function in the class. Dense classes can either be learned in constant space, or they require at least logarithmic space. The intuition is that all dense classes are either trivial and can be learned simple device like a finite automation, or to learn them requires counting some feature of the data. Counting requires logarithmic space. We have examples of dense classes that can be learned in constant space and examples where at least logarithmic space is required for learning.

The order of the input is crucial, as we have an example where learning can be accomplished within a small constant amount of space, if the data arrives in a nice order, but takes linear (the maximum) if the data arrives

in a malicious order.

References

[ACM, 1992] ACM. *Proceedings of the 1992 Workshop on Computational Learning Theory*, New York, NY., 1992. ACM Press.

[ACM, 1993] ACM. *Proceedings of the 1993 Workshop on Computational Learning Theory*, New York, NY., 1993. ACM Press.

[ACM, 1994] ACM. *Proceedings of the 1994 Workshop on Computational Learning Theory*, New York, NY., 1994. ACM Press.

[Anderson, 1983] John Robert Anderson. *The Architecture of Cognition.* Harvard University Press, Cambridge, MA, 1983.

[Angluin and Smith, 1983] D. Angluin and C. H. Smith. Inductive inference: Theory and methods. *Computing Surveys*, 15:237–269, 1983.

[Angluin and Smith, 1987] D. Angluin and C. H. Smith. Inductive inference. In S. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 409–418. John Wiley and Sons Inc., 1987.

[Blum and Blum, 1975] L. Blum and M. Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.

[Case and Smith, 1983] J. Case and C. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25(2):193–220, 1983.

[Freivalds and Smith, 1992] R. Freivalds and C. Smith. Memory limited inductive inference machines. In *Lecture Notes in Computer Science Vol. 621*, pages 19–29. Springer-Verlag, 1992.

[Freivalds *et al.*, 1993] R. Freivalds, E. Kinber, and C. Smith. The impact of forgetting on learning machines. In *Proceedings of the Workshop on Computational Learning Theory*, pages 165–174. ACM, 1993.

[Fulk and Case, 1990] M. Fulk and J. Case, editors. *Proceedings of the Third Annual Workshop on Computational Learning Theory*, Palo Alto, CA., 1990. Morgan Kaufmann Publishers.

[Gold, 1967] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[Haussler and Pitt, 1988] D. Haussler and L. Pitt, editors. *Proceedings of the 1988 Workshop on Computational Learning Theory*, Palo Alto, CA., 1988. Morgan Kaufmann Publishers.

[Levine, 1991] D. Levine. *Introduction to Neural and Cognitive Modeling.* Lawrence Earlbaum Associates, 1991.

[Lewis and Papadimitriou, 1981] H. Lewis and C. Papadimitriou. *Elements of the Theory of Computation.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981.

[Michalski *et al.*, 1983] R. Michalski, J. Carbonell, and T. Mitchell. *Machine Learning.* Tioga Publishing Co., Palo Alto, CA, 1983.

[Miller, 1956] G. Miller. The magical number seven plus or minus two. *The Psychological Review*, 63:81–97, 1956.

[Osherson *et al.*, 1986] D. Osherson, M. Stob, and S. Weinstein. *Systems that Learn.* MIT Press, Cambridge, Mass., 1986.

[Popper, 1968] K. Popper. *The Logic of Scientific Discovery.* Harper Torch Books, N.Y., 1968.

[Rivest *et al.*, 1989] R. Rivest, D. Haussler, and M. Warmuth, editors. *Proceedings of the Second Annual Workshop on Computational Learning Theory*, Palo Alto, CA., 1989. Morgan Kaufmann Publishers.

[Rosenbloom *et al.*, 1991] P. Rosenbloom, J. Laird, A. Newell, and R. McCarl. A preliminary analysis of the soar architecture as a basis for general intelligence. *Artificial Intelligence*, 47:289–325, 1991.

[Servan-Schreiber, 1991] E. Servan-Schreiber. *The Competitive Chunking Theory: Models of Perception, Learning, and Memory.* PhD thesis, Carnegie Mellon University, 1991. Ph.D. thesis, Department of Psychology.

[Shapiro, 1987] S. Shapiro. *Encyclopedia of Artificial Intelligence.* John Wiley and Sons Inc., New York, NY, 1987.

[Soare, 1987] R. I. Soare. *Recursively Enumerable Sets and Degrees.* Springer Verlag, New York, 1987.

[Warmuth and Valiant, 1991] M. Warmuth and L. Valiant, editors. *Proceedings of the 1991 Workshop on Computational Learning Theory*, Palo Alto, CA., 1991. Morgan Kaufmann Publishers.