DRAFT — June 11, 1997

Recursion Theoretic Models of Learning: Some Results and Intuitions[†]

by

Carl H. Smith and William I. Gasarch
Department of Computer Science and
Institute for Advanced Computer Studies
The University of Maryland
College Park, MD 20742, USA

[†] Supported by NSF grants CCR 8701104 and CCR 8803641.

I. An Abstract View of Learning

To implement a program that somehow "learns" it is neccessary to fix a set of concepts to be learned and develop a representation for the concepts and examples of the concepts. In order to investigate general properties of machine learning it is neccesary to work in as representation independent fashion as possible. In this work, we consider machines that learn programs for recursive functions. Several authors have argued that such studies are general enough to include a wide array of learning situations [2,3,22,23,24].

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. Some behaviors, such as anger, are not easily modeled as functions. Our primary interest, however, concerns the learning of fundamental behaviors such as reading (mapping symbols to sounds), recognition (mapping patterns to descriptions), arithmetic (mapping formulas to numbers) and a variety of physical skills. 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.

For the purposes of a mathematical treatment of learning, it suffices to consider only the learning of functions from natural numbers to natural numbers. We will consider a variety of models of learning recursive functions, 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. 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 theorems we cite about these models indicate that, in some ways, machine learning is similar to human learning. To ease notation, we let the natural numbers ($\mathbb{N}$) serve as names for programs.

II. The Gold Model

Gold, in a seminal paper [24], 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, (see Figure 1). 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. This seems to be analogous with human learning. Science is full of surprises. The great breakthroughs occur when somone points out that our thoughts about some phenomenon are not accurate. 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.

Following Gold, we say that an IIM *identifies* a function f, if, when the IIM 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 of some (undecidable) set of recursive functions. The collection of all such sets, over the universe of effective algorithms serving 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.

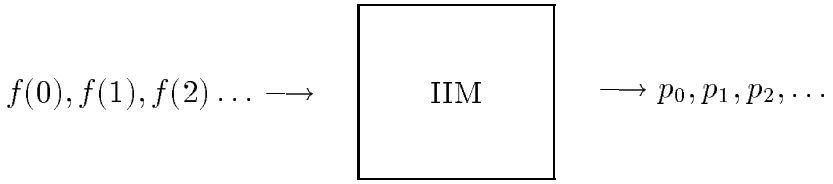

Figure 1 The Gold Model

Gold was interested primarily on *language* learning. A fundamental difference between learning languages and functions is that for language learning, negative information helps while it is of no use what so ever in trying to learn a function f if you know that $f(3) \neq 6$. However, Gold did show the following:

THEOREM 1. (Gold [24]) No inductive inference machine can learn all the recursive functions.

By anthropomorphising the IIMs and applying the theorem to humans no surprises result. We can say, for example, that not only does no one know it all, no one can learn it all. This, perhaps, suggests the specialization that scientists have tended toward in order to learn about the universe we live in.

Since no IIM can learn everything, there is the neccesary mathematical lee way to compare various notions of learning. Some of the other notions envolve variations on the definition of what it takes for an IIM to learn, see [2,3]. The approach taken here is to examine fundamentally different models of learning, with each model geared to focus on a particular aspect of learning.

III. Team Learning

Given that no one IIM can learn all the recursive functions, it is mathematically natural to ask if two, three, or any number of IIMs can. There is also a philosophical motivation for considering teams of IIMs. The first analytical treatise of learning by example can be found in the musings of the philosphers of science [0,6,5,11,12,13,15]. The Gold model can be viewed as an abstraction of the scientific method where the input data represents encodings of experimental results and the output programs represent encodings of explanations of the phenomenon under investigation. Correct explantation can be used to predict the results of, as of yet unperformed, experiments. Scientists rarely work in isolation. Often, there will be several scientists performing experiments in an attempt to understand some given phenomenon. Each scientist will have available, in due time,

the results of all the experiments conducted by all the scientists investigating the same phenomenon. even though they all have the same data, they may each draw different conclusions from them. Typically, there is more than one competing theory. Even generally accepted theories have critics who dispute the truth fo conventional wisdom.

A *team* of IIMs learns a recursive function f iff one of the team members infers f in the sense of the Gold model. Each member of the team sees the same input data and outputs its own sequence of conjectured programs (see Figure 2). As with teams of scientists, there is no effective way of determining which, if any, of the scientists is expousing the correct hypothesis. If one team member learns the input function, the others are not even obliged to converge. Each team will learn a certain set of recursive functions. For each $n \in \mathbb{N}$, the sets of recursive functions learnable by as team of exactly n IIMs will be collected for mathematical scrutiny.

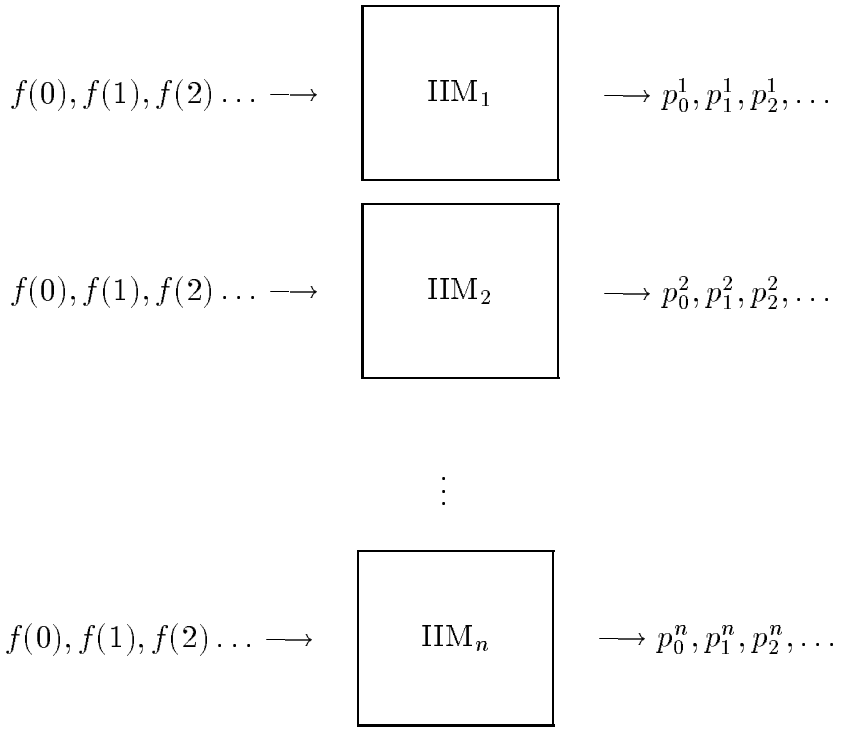

Figure 2 Team Learning

The first result about team inference was the following.

THEOREM 2. (L. Blum and M. Blum [22]) Teams of two IIMs can learn everything single IIMs can learn, and more.

Considering and arbitary size team leads to the following.

THEOREM 3. (Smith [17]) For any $n \in \mathbb{N}$, teams of $n+1$ IIMs can learn everything teams of n IIMs can, and more.

As an immediate consequence of the above result we have the following.

COROLLARY 4. (Smith [17]) No finite collections of IIMs can learn all the recursive functions.

Suppose M_1, M_2, ..., M_n is a team of n IIMs. Let $\hat{M}$ be an IIM with an n-sided coin. To start its operation, $\hat{M}$ flips its coin to choose a number i ($1 \leq i \leq n$). $\hat{M}$ proceeds to simulate M_i on the data it inputs. In this fashion, $\hat{M}$ *probabilistically* learns every recursive function that the team could. Pitt [18] formalized the notion of probabilistic inference and found the deep relationship between team learning and probabilistic learning embodied in the following.

THEOREM 5. (Pitt [18]) For any $n \in \mathbb{N}$ with $n > 0$, for any probability p with $1/(n+1) < p \leq 1/n$, a set S of recursive functions is learnable by a team of n IIMs if and only if S is probabilstically learnable with probability p.

So we see that, in some sense, probabilistic learning is the same as team inference. By anthropomorphising the IIMs as teams of scientists, we see that the "concensus" theory is only correct with some probability. This differs from the human situation only in that it is possible to determine the probability analytically. Questions concerning teams of probabilistic IIMs were raised and answered in [0].

IV. Learning Sequences of Phenomena

The approaches of the previous sections could be termed *tabla rasta* in that the IIMs always start from the same state. Once an IIM has learned (or given up on) some function, no experience gained in the learning effort is carried over to the next learning task. When actual learning programs are implemented, the intention is for them to attempt several learning tasks. People do not start over again every time they want to learn something new. We all *accumulate* knowledge. The existence of "prerequisites" in college course offering structures is indicative of a concensus that there is a prefered order to learning some sets of phenomena. Work on the Soar project [10] evidenced the fact that find some learning tasks easier when some other concept is mastered in advance.

A technique for embuing IIMs with knowledge gained from prior learning efforts was formalized in [19]. We now describe the operation of a *Sequence Inference Machine* (SIM) M. For M to learn the sequence of functions f^1, f^2, ..., f^n is must perform n separate executions. First, M infers f^1 just as an IIM would in the Gold model. Suppose e_1 is a program for f^1. The second run of M takes e_1 as an input and then receives the function f^2 and learns it in the usual manner. The i^{th} run of M starts with a preamble of e_1, e_2, ..., e_{i-1} of programs for f^1, f^2, ..., f^{i-1}, respectively, and then infers f^i as in the Gold model, see Figure 3. For M to succeed at learning the sequence, it must learn each function in the sequence, in the standard way, except that it is given the answers for all prior functions in the sequence. The idea is that some "trainer" tells the SIM when it has finished learning each function in the sequence so that the SIM can start on the next one. The assumption is that, without a noncomputable intervention, the SIM must be getting the preamble of answers from its own prior learning experiences.

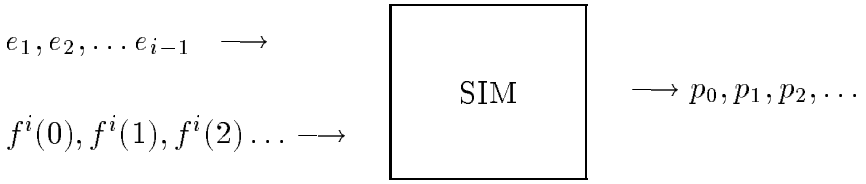

Figure 3 Learning Sequences of Functions

The main result presented below is that, sometimes, prerequisites are absolutely essential. That is, some concepts are sufficiently difficult, that one must first learn something else. For example, the notion of "function" must be mastered before calculus can be attempted. Even more fundamental, the concept of "sitting" must be understood before the idea of chair makes any sense.

THEOREM 6. (Angluin, Gasarch, Smith [19]) For each n, there are sets of length n sequences of functions that can be learned, but *only* in the designated order.

V. Learning Several Phenomena at the Same Time

Few people have the concentration to focus on learning a single phenomenon exclusively. For example, the process of learning a spoken language continues throughout a person's childhood, and often longer. Intersperses with language learning is the learning of several other concepts, both academic and cultural. Many college curricula feature corequisites as well as prerequisites.

Our next model, the *parallel inference machine* (PIM) was also defined in [19]. M a PIM behaves just like an IIM except that it simultaneously inputs data from n different functions and instead of outputting a single sequence of conjectured programs, it outputs n such sequences, one for each input function, see Figure 4. A PIM M learns an n-tuple of recursive functions $\langle f^1, f^2, \ldots, f^n \rangle$ if, for each $1 \leq i \leq n$, M outputs a sequence of programs that converge to e^i where program e^i computes f^i.

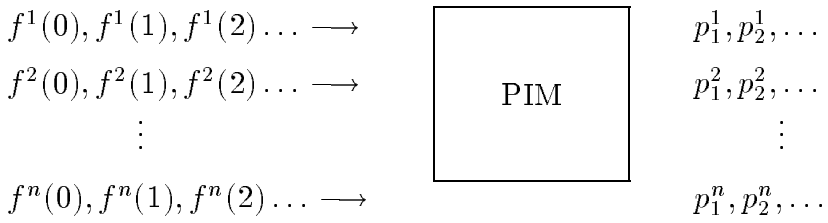

Figure 4 Learning Functions in Parallel

For a given n and an PIM M, M will learn a set of n-tuples of recursive functions. It turns out that everything that can be learned in proper sequence order, can also be learned in parallel and that there are some things that can be learned in parellel that can't be learned by sequence learners.

THEOREM 7. (Angluin, Gasarch, Smith [19]) For every SIM M there is a PIM M' such that if M learns the sequence f^1, f^2, ..., f^n then M' learns the n-tuple $\langle f^1, f^2, \ldots, f^n \rangle$.

THEOREM 8. (Angluin, Gasarch, Smith [19]) For every $n > 0$, there is a set of n-tuples of recursive functions that can be learned in parallel by a PIM such that no SIM can learn all the n-tuples taken as length n sequences.

The above results imply that there are (sets of) pairs of functions f^1 and f^2 that can be learned together, but not individually. Is it the case that ther are (sets of) triples of functions f^1, f^2 and f^3 that are collectively learnable but *any* pair of them is not? The answer is yes, but to state the general result we must have a way of comparing sets of n tuples of functions with sets of m tuples of functions for $m \neq n$. Suppose $n > m > 0$.and An m-projection of the tuple $\langle f^1, f^2, \ldots, f^n \rangle$ is some selection of exactly m functions. For example, perhaps the first m functions, or the last m, or the first $m-1$ and the last one, etc. Suppose S is a set of n tuples of recursive functions. An m-projections of S is a collection of m tuples formed by taking the *same* m-projection of every n-tuple in S.

THEOREM 9. (Angluin, Gasarch, Smith [19]) For any $n > m > 0$ there is a set of n-tuples of recursive functions S, learnable by a PIM, such that no m-projection of S is.

VI. Learning by Asking Questions

Inquisitiveness is the driving force behind all of science. Formal learning situations most always have opportunities for the learner(s) to ask questions. Yet, all of the models

of learning discussed above are *passive* in the sense that they require the learning device to wait and receive data. We proceed to define mechanisms that learn by asking questions. These machines will not input any data except the answers to specific questions. This will not be overly restrictive since the value of $f(x)$, for any x, can be discovered by asking "Is $f(x) = 0$?" "Is $f(x) = 1$?" "Is $f(x) = 2$?" ... until a YES answer is recieved.

A *query inference machine* (QIM) is an algorithmic device that asks a teacher questions about some unknown function, and while doing so, outputs programs. In this way, the QIM learns about some phenomenon, encoded by a recursive function, by asking finitely many questions. We assume that the teacher always returns the correct answer to any question asked by a QIM. The questions are formulated in some query language L. Formally, a QIM is a total algorithmic device which, if the input is a string of bits $\vec{b}$, corresponding to the answers to previous queries, outputs an ordered pair consisting of a guess, a program e, (possibly null) and a question ψ (See Figure 5). A QIM M learns a recursive function f if, when the teacher answers M's questions about f truthfully, the sequence of output programs converges to a program that computes f.

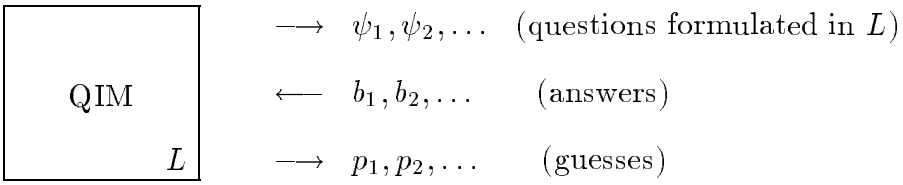

Figure 5 Learning via Queries

A variety of different query languages are considered. The languages that we consider have different expressive power. The more expressive the query language, the more questions the QIM can ask. In a sense, giving a QIM a more expressive query language to use makes the QIM more articulate. Every language allows the use of $\wedge$, $\neg$, $=$, $\forall$, $\exists$, symbols for the natural numbers (members of $\mathbb{N}$), variables that range over $\mathbb{N}$, and a single function symbol $\mathcal{F}$ which will be used to represent the function being learned. Inclusion of these symbols in every language will be implicit. The *base* language contains only these

symbols. Restricting the applications of quantifiers is a technique that we will use to regulate the expressive power of a language. By convention, all questions are assumed to be in prenex normal form (quantifiers followed by a quantifier-free formula, called the matrix of the formula) and questions containing quantifiers are assumed to begin with an existential quantifier. This convention entails no loss of generality.

THEOREM 10. (Gasarch, Smith [20]) Let L be the base language without quantifiers augmented with any effectively computable operators. Then a set S of recursive functions can be learned by a QIM using L as a query language if and only if S can be learned by an IIM (Gold model).

THEOREM 11. (Gasarch, Smith [20]) Let L be the base language with operators for addition and multiplication. There is a QIM using L as a query language that can learn all the recursive functions.

THEOREM 12. (Gasarch, Smith [20]) Let L be the base language with operators for successor and less than. No QIM using L as a query language that can learn all the recursive functions.

Let L be the base language with operators for addition and less than. This language L ios the focus of the last three results of this section. Let L_0 be L without quantifiers. Similarly, let L_1 be L restricted to existential quantifiers only. Finally, let L_2 denote L restricted to questions involving a single alternation of quantifiers. Clearly, QIM's using L_2 as a query language can learn all that QIM's using L_1 can. Similarly, QIM's using L_1 as a query language can learn all that QIM's using L_0 can

THEOREM 13. (Gasarch, Smith [20]) QIM's using L_1 as a query language can learn strictly more that QIM's using L_0 can.

THEOREM 14. (Gasarch, Kinber, Pleszkoch, Smith, Zeugmann [0]) QIM's using L_2 as a query language can learn strictly more that QIM's using L_1 can.

THEOREM 15. (Gasarch, Pleszkoch, Solovay [8]) No QIM using L as a query langauge can learn all the recursive functions.

VII. Procrastnation and Learning

A phrase we have all heard at least once in our lives is "let me sleep on it." While the body sleeps, the mind sorts through the events of the day. The source of the common expression is perhaps some subconscience, or perhaps conscience, desire to procrastinate about making a decision. Procrastination is common in academia as witnessed by the length of time some papers spend with editors and referees. One view is that procrastination is a natural by product of the contemplative academic life style. Another view puts procrastination as the cause. We all display a tendency to delay (or sometimes avoid) performing unpleasant tasks. While procrastination is generally regarded as a undesirable quality, the results presented below point out a potential advantage to some uses of procrastination.

For practical applications, one cannot wait "until the limit" for an answer. Hence, IIMs constraint to change their output conjecture a predetermined number of times were investigated in [23]. For example. referring to Figure 1, if an IIM outputs p_0 then $p_1 \neq p_0$ and then p_0 again, it has changed its conjecture twice. For each $n \in \mathbb{N}$ there is a collection of sets, each one of which can be learned by some IIM constrained to produce at most n changes of conjecture. Allowing more trials (changes of conjecture) enlarges the collection of sets of functions that become learnable.

THEOREM 16. (Case, Smith [23]) For each $n \in \mathbb{N}$, the collection of sets of recrusive functions that are learnable by IIMs constrained to produce at most n changes of conjecture is strictly included in the corresponding collection for $n+1$. Furhtermore, each of these constrained collections is strictly included in the collections of sets learnable via the Gold model.

Now we consider inference devices that *procrastinate* about how many trials they can make. The idea is to let the device announce that it will succeed (stop) after, say 5, changes of conjecture. At the point of the 5^{th} change of conjecture, the device has the option of declaring that another, say 7, trials are needed. The recursive (constructive) ordinals [0,14] are used to denote various procrastination stratgies.

We now proceed to define the operation of an *ordinal inductive inference machine* (OIM). An OIM is given a recursive ordinal α to specify the maximum number of mind changes. For the sake of example, let $\alpha = \omega^{\omega}$. Associated with the ordinal is a counter, c_{α}, initialized to 0. Like an ordinary IIM, the OIM reads input data and makes conjectures. In addition the OIM, from time to time, *reduces* the ordinal α and *increases* the counter c_{α}, see Figure 6. For each recursive ordinal there is a collection of sets such that each set in the collection is learnable by some OIM using the selected ordinal. The OIM will, by assumption, reduce the ordinal sufficiently often so as to diminish its value to 0.

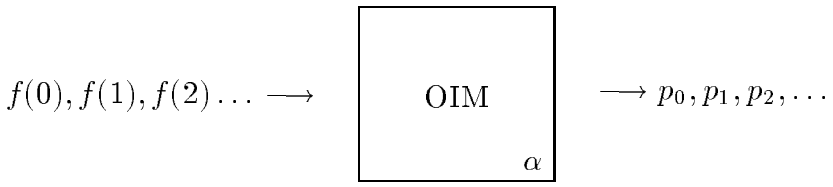

Figure 6 Learning with Procrastination

The OIM reduces its ordinal, and updates the associated counter, by requesting a *modification* operation. The modification algorithm is relatively complex. Form the standpoint of the OIM, it looks like a simple command. The modification operation has three cases. These are described briefly before returning to our particular example.

Case 1. The ordinal is 0. No modifications are done, the counter and ordinal remain the same.

Case 2. The ordinal is a limit ordinal. Then the rightmost, outermost ω term of the ordinal is replaced by a (finite) member of ω and the associated counter is incremented by 1. The choice of the member of ω is made deterministically by the OIM.

Case 3. The ordinal is a successor ordinal. The ordinal is reduced by one, e.g. replace the successor ordinal by the ordinal which it follows, and the counter is incremented by 1.

Returning to our example, $\alpha = \omega^\omega$ is a limit ordinal, so Case 2 applies to the first modification. To keep the notation simple, suppose that the OIM chooses $2 \in \omega$. After the first modification, $c_\alpha = 1$ and $\alpha = \omega^2 = \omega \cdot \omega$. Since, α is still a limit ordinal, Case 2 applies to the next modification. Suppose this time the OIM chooses $3 \in \omega$. Then c_α becomes 2 and α becomes $\omega \cdot 3 = \omega + \omega + \omega$. Again, the modified α is still a limit ordinal, so Case 2 applies for the next modification. If the OIM chooses 4, then $c_\alpha = 3$ and $\alpha = \omega + \omega + 4$.

At this point, α is now a successor ordinal, so Case 3 applies for the first time in this example. In fact, Case 3 will apply four times in a row. The modifications performed are given in the table below where moving from one line to line immediately beneath represents a single modification.

$\alpha = \omega + \omega + 4$	$c_\alpha = 3$
$\omega + \omega + 3$	4
$\omega + \omega + 2$	5
$\omega + \omega + 1$	6
$\omega + \omega + 0$	7

Since $\omega + \omega + 0 = \omega + \omega$, α is once again a limit ordinal, so Case 2 applies once again. Suppose 5 is chosen by the OIM making $\alpha = \omega + 5$ and $c_\alpha = 8$. Five more consecutive modifications according to Case 3 yields $\alpha = \omega$ and $c_\alpha = 13$.

For the last time in this example, α is a limit ordinal. Suppose the final choice by the OIM is 6. Then α becomes 6 and c_α becomes 14. Six more modifications via Case 3 results in $\alpha = 0$ and $c_\alpha = 20$. All further requests for modification by the OIM will be handled by Case 1 and the values of α and c_α will not change.

THEOREM 17. (Frevalds, Smith [21]) For any recursive ordinals $\alpha > \beta$, The collections of sets of recursive functions that are learnable by OIMs using α to determine how many mind changes to make includes and is strictly larger than the analogous class derived from β.

COROLLARY 18. (Frevalds, Smith [21])

No OIM can learn all the recursive functions.

THEOREM 19. (Frevalds, Smith [21]) The collection of learnable sets, over all OIMs and over all recursive ordinals is strictly smaller than the collection of sets learnable via the Gold model

References

1. ANGLUIN, D., GASARCH, W. I., AND SMITH, C. H. Training sequences. *Theoretical Computer Science 66* (1989), 255–272.
2. ANGLUIN, D. AND SMITH, C. H. Inductive inference: theory and methods. *Computing Surveys 15* (1983), 237–269.
3. ANGLUIN, D. AND SMITH, C. H. Inductive inference. In *Encyclopedia of Artificial Intelligence*, S. Shapiro, Ed., John Wiley and Sons Inc., 1987.
4. BLUM, L. AND BLUM, M. Toward a mathematical theory of inductive inference. *Information and Control 28* (1975), 125–155.
5. BURKS, A. W. *Collected Papers of Charles Sanders Peirce.* Harvard University Press, Cambridge, Mass., 1958.
6. CARNAP, R. AND JEFFREY, R. *Studies in inductive logic and probability.* University of California Press, Berkeley, California, 1971.

7. Case, J. and Smith, C. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science 25,* 2 (1983), 193–220.
8. Gasarch, W., Pleszkoch, M., and Solovay, R. *Learning via queries with plus and times.* manuscript.
9. Gold, E. M. Language identification in the limit. *Information and Control 10* (1967), 447–474.
10. Laird, J., Rosenbloom, P., and Newell, A. Chunking in Soar: the anatomy of a general learning mechanism. *Machine Learning 1,* 1 (1986), 11–46.
11. Popper, K. *The Logic of Scientific Discovery.* Harper Torch Books, N.Y., 1968. 2nd Edition.
12. Putnam, H. Probability and confirmation. In *Mathematics, Matter and Method*, 1, Cambridge University Press, 1975. Originally appeared in 1963 as a Voice of America Lecture.
13. Rescher, N. *Scientific Explanation.* The Free Press, New York, 1970.
14. Rogers, H. Jr. *Theory of Recursive Functions and Effective Computability.* McGraw Hill, New York, 1967.
15. Schilpp, P. *Library of Living Philosophers: The Philosopy of Rudolph Carnap.* Open Court Publishing Co., LaSalle, IL, 1963.
16. Smith, C. and Freivalds, R. *On the power of procrastination for machine learning.* Manuscript.
17. Pitt, L. A Characterization of Probabilistic Inference. *Journal of the ACM 36,* 2 (1989), 383–433.
18. Pitt, L. and Smith, C. Probability and plurality for aggregations of learning machines. *Information and Computation 77* (1988), 77–92.
19. Gasarch, W. and Smith, C. Learning via Queries. *Journal of the ACM.* To appear.
20. Gasarch, W., Kinber, E., Pleszkoch, M., Smith, C., and Zeugmann, T. *Learning via queries with teams and anomalies.* Manuscript.
22. Smith, C. H. The power of pluralism for automatic program synthesis. *Journal of the ACM 29,* 4 (1982), 1144–1165.
23. Church, A. The constructive second number class. *Bulliten of the AMS 44* (1938), 224–232.
24. Carnap, R. *The Continuum of Inductive Methods.* The University of Chicago Press, Chicago, Illinois, 1952.