COMPUTING IN THE LIMIT

ANTONY VAN DER MUDE

Abstract. We define a class of functions termed "Computable in the Limit", based on the Machine Learning paradigm of "Identification in the Limit". A function is Computable in the Limit if it defines a property P_p of a recursively enumerable class A of recursively enumerable data sequences $\vec{S} \in A$, such that each data sequence $\vec{S}$ is generated by a total recursive function ϕ_s that enumerates $\vec{S}$. Let the index s represent the data sequence $\vec{S}$. The property $P_p(s) = x$ is computed by a partial recursive function $\phi_p(s,t)$ such that there exists a u where $\phi_p(s,u) = x$ and for all $t \geq u$, $\phi_p(s,t) = x$ if it converges. Since the index s is known, this is not an identification problem - instead it is computing a common property of the sequences in A. We give a Normal Form Theorem for properties that are Computable in the Limit, similar to Kleene's Normal Form Theorem. We also give some examples of sets that are Computable in the Limit, and derive some properties of Canonical and Complexity Bound Enumerations of classes of total functions, and show that no full enumeration of all indices of Turing machines TM_i that compute a given total function $f(x)$ can be Computable in the Limit.

1. Introduction

There has been a considerable amount of work on hypercomputation - functions that can be formally defined but are beyond the capability of any Turing machine[4]. One well-known example of a hypercomputation is that of Incompressible Numbers from Kolmogorov Complexity[3]. Kolmogorov Complexity is the function $K(x) = y$, where Turing machine TM_y is the machine with the shortest index y that, starting with an empty tape, halts with x on its tape. The Incompressible Numbers are numbers where, essentially, $K(x) = |x| + c$. This happens when there is no shorter way of generating these numbers except by storing a copy of the number x in the state table of the Turing machine, which is just the size of x plus a constant c. These values are known to be immune - no infinite recursively enumerable set is a subset of the Incompressible Numbers.

Sets such as K have a conceptual genesis in Machine Learning. Kolmogorov based his complexity theory on work by Ray Solomonoff[6][7]. Solomonoff, and later Mark Gold[2], developed some initial results on a type of Machine Learning called Identification in the Limit. Identification in the Limit is a learning task where the learner is presented with a sequence of elements from a recursively enumerable set and is given the task to determine an index i for the function ϕ_i whose domain is the set.

In this paper, we will consider a more general problem. We will start with the data set and its identification ϕ_i and try to discover properties of this data. This is not an identification task - we know its identity already, but we want to compute interesting characteristics of the data. In the case of Kolmogorov complexity, the

data set is a single value x and the property is the shortest function that computes that value.

The properties that we are looking for are computed in the limit - we make a guess, or find an initial result and update our hypothesis as we go along. This yields a nice normal form to characterize this class of Computable in the Limit tasks, similar to the Kleene Normal Form Theorem. We will also explore some properties that are Computable in the Limit and some that are not.

2. Definitions

Using the notion of Rogers[5] we define the Tau function as

$$\tau(x,y) = \frac{1}{2}(x^2 + 2xy + y^2 + 3x + y)$$

Where

$$\tau^1(x) = x$$
$$\tau^2(x,y) = \tau(x,y)$$
$$\tau^{k+1}(x_1,...,x_{k+1}) = \tau(\tau^k(x_1,...,x_k), x_{k+1})$$

and $\pi_i^k(x)$ is the inverse function for the i^{th} element of τ^k.

Since we are dealing with functions that are given as an input an initial segment of an infinite sequence, we assume that a function can be given a sequence of variables $\langle S(0),...,S(n)\rangle$ as an encoded input x using the tau function

$$\vec{S} = \langle S(0),...,S(n)\rangle = \tau^{n+2}(n, S(0),...,S(n)) = x$$

We use the notation TM_i for the Turing machine with state table i. $TM_i(0)$ is the computation on an empty tape. $TM_i(x)$ is the computation on input x. An equivalent notation is ϕ_i, $\phi_i(0)$ and $\phi_i(x)$, which is the partial recursive function equivalent to the Turing machine TM_i. The time complexity on input x is given as $\Phi_i(x)$, which is the number of steps that $TM_i(x)$ takes until it converges. A function with multiple inputs, for example $\phi_i(x,y,z)$, is computed by a Turing Machine $TM_i(x,y,z) = TM_i(\tau^3(x,y,z))$.

If a set A is enumerable by a recursive function ϕ_i, then ϕ_i is a total function whose range is exactly A.

We begin with a very general definition of Computing in the Limit for a single data sequence, where this input data sequence can be anything.

Definition 2.1. A property P_p is **Computable in the Limit on an Sequence** $\vec{S}$, where $\vec{S}$ is an infinite sequence of inputs $\vec{S} = \langle (S(0),...,S(n),...\rangle$, and $P_p = \phi_q$ is a partial recursive function whose inputs are initial segments of $\vec{S}$, where there exists a u such that $\phi_q(\langle S(0),...,S(u)\rangle) = x$ and for all $t > u$, we have $\phi_q(\langle S(0),...,S(t)\rangle) = x$ if it converges. If this happens, we say P_p **Converges on** $\vec{S}$ **in the Limit, or *l*-converges to** x.

In this case, $P_p(\vec{S}) = x$, that is, P_p computes x in the limit as a property of the sequence $\vec{S}$. If there is no final value x, P_p changes its mind an infinite number of times and therefore P_p ***l*-diverges**. Note that P_p can be partial, so that $\phi_q(\langle S(0),...,S(n)\rangle)$ can fail to halt for some arbitrary (even infinite) number of values n. We use the notation $P_p(\vec{S}) = x$ to indicate that $P_p(\vec{S})$ is **Computable in the Limit on** $\vec{S}$ **as** x.

So the Computing in the Limit procedure must be a partial recursive function. But since the input data sequence can be anything at all, the problem is hard to characterize. For the sake of tractability, we will assume that the sequences are recursively enumerable: the sequence $\vec{S}$ is the output of a total recursive function ϕ_s.

Lemma 2.2. *If $\vec{S}$ is recursively enumerable by a total function $\phi_s(n) = S(n)$ then $P_p(\vec{S})$ is a partial recursive function $\phi_p(s,t)$ where, if $P_p(\vec{S})$ l-converges to x then there exists a u such that $\phi_p(s,u) = x$ and for all $t > u$ if $\phi_p(s,t)$ converges, then $\phi_p(s,t) = x$.*

Proof. Trivial. $\phi_p(s,n) = \phi_q(\phi_r(s,n))$ where

$$\phi_r(s,n) = \tau^{n+2}(n, \phi_s(0), ..., \phi_s(n))$$

since the $S(n)$ values are generated by the function ϕ_s. □

If $\vec{S}$ is the output of a total recursive function ϕ_s then we can replace the sequence in the input of the property $P_p(\vec{S})$ with the index of the total recursive function that enumerates it: $P_p(s)$.

Definition 2.3. If $\vec{S}$ is recursively enumerable by a total function $\phi_s(n) = S(n)$, and $P_p(\vec{S})$ is Computable in the Limit for that Sequence $\vec{S}$, then we use the notation $\boldsymbol{P_p(s) = x}$ where P_p is a partial recursive function

$$\phi_p(s,y) = \phi_q(\langle \phi_s(0), ..., \phi_s(y) \rangle)$$

where there exists a u such that $\phi_p(s,u) = x$ and for all $t > u$ if $\phi_p(s,t)$ converges, then $\phi_p(s,t) = x$. We say that $\boldsymbol{P_p(s)}$**is computed by** $\boldsymbol{\phi_p(s,t)}$.

The properties that are Computable in the Limit can apply to a whole class of sequences that have the same property P. If the class of sequences is a recursively enumerable class of recursively enumerable sequences, then we can replace the sequence in $P(\vec{S})$ with $P(s)$, where for all n, $\phi_s(n) = S(n)$.

Since we know the index s of a generating function for the data set, we are not identifying the set itself, but instead we are computing properties of the set. That is why we use the term "Computing in the Limit".

Definition 2.4. Assume that A is an infinite recursively enumerable class of recursively enumerable sequences $\vec{S}$, where the range of total function ϕ_a are values s such that if $\vec{S}_s \in A$ then for all n, $\phi_s(n) = S_s(n)$. Then for all $\phi_a(n) = s$, $P(s) = y$ for some y or $P(s)$ l-diverges. Let property P_p be computed by ϕ_p for every sequence in A. This property $P_p(x)$ over A is **Computable in the Limit**. The property P_p is ***l*-Total** if P_p never l-diverges on any input sequence in A. Otherwise P_p is ***l*-Partial**.

Note that the set of all shortest algorithmic descriptions of Kolmogorov complexity is expressed by a such property P_K. P_K takes the input x and generates a sequence $\langle x, x, x, ... \rangle$ at time n of size n. It is trivial to show that for every value x, there is a Turing machine TM_z defined uniformly on x, that just encodes the digits of x in its state table. Thus, $z = |x| + c$. Therefore any shortest encoding of x must be shorter than z. So for each initial segment of the sequence of size n, $P(x)$ runs $TM_0(0)$ through $TM_z(0)$ each for n steps and outputs y if there exists a value $y < z$ where $TM_y(0)$ halts in less than n steps and outputs x. Otherwise it

outputs z. Then P_K l-converges to z only if x is incompressible. This property is l-total.

The sequences $\langle x, x, x, ...\rangle$ used in this formulation of Kolmogorov complexity are not just a special case. It is trivial to show that given such a class A, for every property $P_p(s)$, $\vec{S}_s \in A$, there is an equivalent class A_Z and an equivalent property $P_{p'}(s')$, $\vec{S}_{s'} \in A_Z$ where the sequence $\vec{S}_{s'}$ is simply $\langle s, s, s, ...\rangle$. We will define a special class composed of these sequences.

Definition 2.5. A_Z is an infinite recursively enumerable class of recursively enumerable sequences $\vec{S}$ where for all x, $\phi_Z(x) = y$, and for all n, $\phi_y(n) = x$. Therefore $\vec{S}_x \in A_Z$ is the sequence $\langle x, x, x, ...\rangle$. The inverse function is simply $\phi_Z^{-1}(y) = \phi_y(0) = x$.

Lemma 2.6. *Assume that $P_p(x)$ over A is Computable in the Limit. Then there is an equivalent property $P_{p'}(y)$ over A_Z where for all x there exists a y such that $P_p(x) = P_{p'}(y)$.*

Proof. Trivial. Let $P_p(x)$ over A be computed by

$$\phi_p(x, t) = \phi_q(\langle \phi_x(0), ..., \phi_x(t)\rangle)$$

Let $\phi_Z^{-1}(y) = x$. Define $P_{p'}(y)$ over A_Z to be computed by

$$\phi_{p'}(y, t) = \phi_{q'}(\langle \phi_y(0), ..., \phi_y(t)\rangle) = \phi_{q'}(\langle x, ..., x\rangle) = \phi_q(\langle \phi_x(0), ..., \phi_x(t)\rangle) = \phi_p(x, t)$$

□

From this point on, we will assume that all properties P_p are over A_Z.

3. The Normal Form Theorem

The notation $P_p(s)$ for properties Computable in the Limit looks similar to the standard definition of a partial recursive function $\phi_p(s)$. Actually the only difference is the terminating condition. This allows us to construct a Normal Form Theorem that is similar to Kleene's Normal Form theorem except that instead of using the mu function $\mu(x)$ to find the first element where a Boolean predicate is true, we use a new function $\lambda(x)$ to find the last of a finite number of guesses.

Definition 3.1. The function $\mu(x)[...x...]$ is the least integer x such that the expression $...x...$ is true when "x" is interpreted as the integer x, if $...x...$ is true at least one point[5].

Definition 3.2. The function $\lambda(x)[...x...]$ is the largest integer x such that the expression $...x...$ is true when "x" is interpreted as the integer x, if $...x...$ is true at a finite number of points.

Theorem 3.3. Kleene's Normal Form Theorem*: There exists a primitive recursive function U and a primitive recursive predicate $T(e, x, y)$ such that every function $f(x)$ is effectively computable iff $f(x) = \phi_e(x) = U(\mu y T(e, x, y))$.*

The Boolean function $T(e, x, y)$ encodes in the variable y the computation history of Turing machine TM_e on input x. The predicate $T(e, x, y)$ returns true if y is a halting sequence. The function U recovers the output from the computation history y. We shall assume that if $T(e, x, y)$ is true then for all $z > y$ where z is y with one or more copies of the last tape configuration of y appended to y, then $T(e, x, z)$ is true also.

Theorem 3.4. Computing in the Limit Normal Form: *There exists a primitive recursive function U and a primitive recursive predicate $T'(e, x, y)$ such that every property $P_p(x)$ is Computable in the Limit iff $P_p(x) = U(\lambda y T'(p, x, y))$.*

Proof. By the definition of $P_p(x)$, there is a $\phi_p(x, n)$ that computes P_p. By definition, $\phi_p(x, n) = \phi_p(\langle x, n\rangle) = TM_p(\tau(x, n))$. By Kleene's Normal Form Theorem there is a U and T such that

$$\phi_p(x, n) = U(\mu y T(p, \tau(x, n), y))$$

Define T' from T as follows (for each p and x):

$T'(p, \tau(x, n), y)$ is true iff $T(p, \tau(x, n), y)$ is true and for all $m < n$ and $z < y$ where $T(p, \tau(x, m), z)$ is true, then $U(y) \neq U(z)$.

Note that this condition does not use the μ function.

Assuming $T(p, x, y)$ and $U(x)$ are primitive recursive, $T'(p, \tau(x, n), y)$ is also.

If the following two conditions are true for some time n:

- Let $\{\langle i_1, y_1\rangle, ..., \langle i_n, y_n\rangle, ...\}$ be the set of all pairs such that for each n, $T(p, \tau(x, i_n), y_n)$ is true.
- For all $m > n$ if $T(p, \tau(x, i_m), y_m)$ is true then $U(y_n) = U(y_m)$

then the following must be true:

If y_i is the largest value in the set $\{y_0, ...y_n\}$, then $T(p, \tau(x, y_i), z)$ is true, where $z > y_i$ and z is a configuration with one or more copies of the last tape configuration in y_i appended to z.

So $U(z)$ is the last value where TM_p changes its mind on the input sequence. Therefore, the last time $T'(p, x, y)$ is true is exactly this value $y = z$.

So $P_p(x) = U(\lambda y T'(p, x, y))$. □

Since this is similar to Kleene's Normal Form Theorem for the general recursive functions except that the μ in the normal form is replaced by λ, we can tighten the Normal Form theorem to be a predicate $T(e, x, y)$ that is true for at most one value y.

Corollary 3.5. Revised Normal Form Theorem for Effectively Computable Functions: *There exists a primitive recursive function U and a primitive recursive predicate $T''(e, x, y)$, that is true for at most one point y for each pair e and x, such that every function $f(x)$ is effectively computable iff*

$$f(x) = \phi_e(x) = U(\mu y T''(e, x, y)) = U(\lambda y T''(e, x, y))$$

Proof. Define $T''(e, x, y)$ to be true iff $T(e, x, y)$ is true and for all $z < y$, $T(e, x, z)$ is false. □

This means that a property that is Computable in the Limit is effectively computable if and only if there is an associated function that makes a single guess and never changes its mind.

4. Some Properties Computable in the Limit

Here are a number of properties that are Computable in the Limit but not effectively computable. These are all pretty obvious, so I shall just state them.

Note that Computable in the Limit properties can be l-partial functions, where for $P_p(x)$ there are either an infinite number of guesses or no guess at all. Some of the examples here are l-total $P_p(x)$ functions. In all of these cases, the class of sequences is A_Z:

- The Kolmorogorov set $K(x) = P_K(x) = y$. The description was given in the Definitions as an example. P_K is l-total.
- Kolmogorov incompressible numbers: $P_I(n) = x$, where x is incompressible. Use $P_K(x)$ and discard any values y that are compressible. When that happens, shift the guesses for $P_I(n)$ down. At each value n there comes a time when all of the previous compressible values are found, and $P_I(n)$ is never shifted. P_I is l-total.
- The set of all partial recursive functions TM_x where $P_P(x) = \langle x, y\rangle$ only if TM_x is partial and y is the smallest value where $TM_x(y)$ diverges. The value $P_P(x, y)$ is computed as follows: for all $z \leq y$ run $TM_x(z)$ for y steps at most. If there is a value z where $TM_x(z)$ has not converged in y steps or less, output $\langle x, z\rangle$. If they all converge, output $\langle x, y\rangle$. If TM_x is total, then if $\Phi_x(y) < y$ for all but a finite y, then $P_P(x, y) = \langle x, y\rangle$ for all but a finite number of cases and does not l-converge. Otherwise, the output of $P_P(x, y)$ changes infinitely often as the smaller inputs converge. In this case, P_P is l-partial.
- The set of all partial recursive functions TM_x where for every n there is a unique x such that $P_Q(n) = x$. Use the same trick as above for the l-total property of Kolmogorov incompressible numbers. In this case, we must keep track of both the function TM_a and the value $P_P(a) = \langle a, b\rangle$ at which it diverges. Ordering the $\langle a, b\rangle$ pairs, $P_Q(n)$ outputs the value c, where $\langle c, d\rangle$ is the n^{th} smallest pair. This takes l-partial P_P and turns it into l-total P_Q.
- The set of all partial functions with finite domain. Instead of keeping track of $\langle a, b\rangle$ where $TM_a(b)$ diverges, keep track of $\langle a, \langle b_0, ..., b_n\rangle\rangle$ where $TM_a(b_i)$ converges, for each $i \leq n$.
- The minimum index for finite sets - similar to the Kolmorogorov set.
- The set of all functions TM_i with a single element in its domain.
- Given an enumeration of indexes of Polynomial functions A and enumeration of indexes of NP functions B, the Boolean property over pairs $\langle i, j\rangle$ where $P_{eq}(\langle i, j\rangle) = 1$ iff $i \in A$, $j \in B$ and $\phi_i = \phi_j$. Otherwise $P_{eq}(\langle i, j\rangle) = 0$. If $i \notin A$ or $j \notin B$, then $P_{eq}(\langle i, j\rangle)$ diverges.
- The property $P_{exp}(i) = \langle i, e\rangle$ where ϕ_i has polynomial complexity with exponent e. If ϕ_i is not polynomial, then $P_{exp}(i)$ l-diverges.

The Polynomial - NP property can be generalized to any pairs of recursively enumerable classes of total functions. We will present two types of generalizations. Let the class of total functions A be enumerated by a total function ϕ_a whose range is indexes of total functions in the set A, and similarly for B and ϕ_b. A theorem by Blum and Blum on the extrapolation of total recursive functions uses the concept of an h-easy function. They show that each recursively enumerable class of functions is bounded, up to a finite number of exceptions, by the computational complexity of a total function h.[1]

Definition 4.1. Let h be a total recursive function. A partial recursive function ϕ_i is **h-easy** if $\Phi(x) \leq h(x)$ for all but a finite number of integers x.

This definition differs slightly from Blum and Blum's in that ϕ_i can diverge in a finite number of cases.

Theorem 4.2. *For any two total recursive functions g and h, P_{eq} is Computable in the Limit, where if ϕ_i is g-easy and ϕ_j is h-easy then $P_{eq}(\langle i,j\rangle) = 1$ if $\phi_i = \phi_j$ and $P_{eq}(\langle i,j\rangle) = 0$ if $\phi_i \neq \phi_j$ and $P_{eq}(\langle i,j\rangle)$ diverges if either ϕ_i is not g-easy or ϕ_j is not h-easy.*

Proof. The function $\phi_{eq}(\langle i,j\rangle, t)$ is computed as follows. For all $x \leq t$, run $\phi_i(x)$ for at most $g(x)$ steps and run $\phi_j(x)$ for at most $h(x)$ steps. If y is the largest value where $\Phi_i(y) > g(y)$ or $\Phi_j(y) > h(y)$ or both, then output $y+1$ unless $y+1$ was output before at some time. Otherwise, output 0 if there is a case z where $\phi_i(z) \neq \phi_j(z)$. If none are found, then output 1.

The output $y+1$ is a guess that ϕ_i is g-easy and ϕ_j is h-easy where all exceptions are less than or equal to y. From that point on, ϕ_{eq} guesses 1 (the two are equal) until an exception is found. □

So the Boolean test of equality for h-easy classes of total recursive functions is Computable in the Limit.

Note, though, that neither the class of Polynomial functions nor the class of NP functions can be defined in this way, because the h-easy function will grow to include all polynomial exponents. This allows for functions whose computation time is not within any exponent. We can, though, enumerate indexes of the polynomial functions of the NP functions, and use them instead of g and h.

Theorem 4.3. *For any two total recursively enumerable sets A and B of total recursive functions, P_{eq} is Computable in the Limit, where if $\phi_i \in A$ and $\phi_j \in B$ then $P_{eq}(\langle i,j\rangle) = 1$ if $\phi_i = \phi_j$ and $P_{eq}(\langle i,j\rangle) = 0$ if $\phi_i \neq \phi_j$ infinitely often, and $P_{eq}(\langle i,j\rangle)$ diverges if either $\phi_i \notin A$ or $\phi_j \notin B$.*

Proof. In this case, $\phi_{eq}(\langle i,j\rangle, t)$ checks that $\phi_i \in A$ and $\phi_j \in B$ before running the test for equality. □

Note that these two theorems do not work for sets of partial recursive functions, because we can get stuck on a case where $\phi_i(x)$ converges and $\phi_j(x)$ diverges. The procedure does not identify this case, so it will return equality even though one function diverged.

It is also possible to Compute in the Limit the ratio of equal to unequal values in the two classes, if the ratio is a rational number bounded in the limit.

Definition 4.4. Given two total functions F and G, the **Error of G on F** (or F on G) is the value $Err(x,F,G) = |\{y \leq x | F(y) \neq G(y)\}| / x$. This is a rational number in the range 0 to 1.

Definition 4.5. The Error of G on F Converges to v if there exists an error bound $\varepsilon(x) = a/b$ such that for each x, the value a/b is a rational number in the range 0 to 1, where $\varepsilon(x)$ is subject to the following two conditions:

- For all u and v, if $u < v$ then $\varepsilon(u) \geq \varepsilon(v)$.
- For all rational numbers c/d in the range 0 to 1 where $c/d \neq 0$ there exists a t where $c/d > \varepsilon(t)$.

Then there exists an m such that for all $n \geq m$ if $Err(n,F,G) = w$ then $|v - w| \leq \varepsilon(n)$.

Theorem 4.6. *For any two recursively enumerable classes of total recursive functions, A and B, enumerable by ϕ_a and ϕ_b, and an error bound $\varepsilon(x)$,*

$$P_{err}(\langle i,j\rangle) = \langle i,j,x,y\rangle$$

is Computable in the Limit, where for some n and m, $\phi_a(n) = i$ and $\phi_b(m) = j$ and the error of ϕ_i on ϕ_j converges to a rational number x/y.

Proof. The function $\phi_{err'}(\langle i, j\rangle, t)$ is computed as follows. Run $\phi_i(x) = \phi_j(x)$ for each $x \leq t$. Compute $Err(t, \phi_i, \phi_j) = w$. Find the rational number a/b with the smallest denominator b such that $|w - a/b| \leq \varepsilon(n)$.

The assumption is that the error of ϕ_i on ϕ_j converges to a rational number x/y. So there is an m such that for all $n \geq m$ if $Err(n, F, G) = w$ then $|w - x/y| \leq \varepsilon(n)$.

If e/f is a rational number where $e/f \neq x/y$ then there exists a rational c/d where $|e/f - x/y| = c/d \neq 0$. So there exists a t where $c/d > \varepsilon(t)$. For all $s \geq t$, $|w - e/f| > \varepsilon(n)$ so e/f is not chosen after time t. This is true for all rational values e/f where $f < y$ so all rationals in the range 0 to 1 are rejected in favor of x/y at some time r. Therefore $\phi_{err'}(\langle i, j\rangle, r) = x/y$ and for all $s > r$, $\phi_{err'}(\langle i, j\rangle, s) = \phi_{err'}(\langle i, j\rangle, r)$. □

Note that, although this theorem does not apply to the reals in general, any real number that can be computed as the output of a Turing machine starting with a blank tape (such as e or π) can be added to this function by using the index of the associated Turing Machine as a possible output.

Given a Computing in the Limit Problem P_p where the output is a class of total functions ϕ_i, we can generate a Canonical listing, where every function appears only once.

Definition 4.7. A **Canonical Enumeration** of total functions is a Computing in the Limit Problem P_p where every function is unique. That is, for any x and y, if $P_p(x) = i$ and $P_p(y) = j$ then there is a z such that $\phi_i(z) \neq \phi_j(z)$.

Theorem 4.8. *Assume a class of total functions is Computable In the Limit by P, in the sense that for each x, if $P(x) = y$, then ϕ_y is a total recursive function. Then there is a* l*-total P' where an index of each function is given exactly once.*

Proof. P' is derived from P where we only add in a function if it differs with each of the previous functions by at least one input. If P changes its mind, we have to recompute the differences. Eventually, each output for P' *l*-converges. □

Definition 4.9. A **Complexity Bound Enumeration** of a canonical enumeration of total functions is an enumeration of all values i where there is a ϕ_j in the canonical enumeration such that

- For all x, $\phi_i(x) = \phi_j(x)$
- For all x, $\Phi_i(x) \leq \Phi_j(x)$

A Complexity Bound Enumeration finds all of the algorithms that are faster than the one in the Canonical Enumeration.

Theorem 4.10. *If a class of total functions is Canonically Enumerable, then the complexity bound enumeration is Computable in the Limit.*

Proof. Given the canonical listing, diagonalize each canonical index i over the enumeration of all Turing machines TM_j, discarding any that differ ($\phi_i(x) \neq \phi_j(x)$) or take too much time ($\Phi_i(x) > \Phi_j(x)$). □

5. Some Properties That Are Not Computable in the Limit

We end with a couple of examples of properties that are not Computable in the Limit.

Definition 5.1. A **Complete Enumeration** of a class of functions A is an l-total property P_p, Computable in the Limit, such that, for all $TM_i \in A$, every function $TM_j = TM_i$ has a value x where $P_p(x) = j$.

Theorem 5.2. *The Complete Enumeration of all Total Recursive Functions is not Computable in the Limit by any* l*-Total* P_p.

Proof. Assume the opposite: the complete enumeration of all total recursive functions is Computable in the Limit by a l-Total P_p.

Since ϕ_p is l-total then for each x there exists a u such that for all $t \geq u$ if $\phi_p(x,t)$ converges, then $\phi_p(x,t) = \phi_p(x,u) = y$ and ϕ_y is a total recursive function. Also, every index j of a total function TM_j is an output of $P_p(v)$ for some v.

Define a family of functions $f(i)$ where for each i, $\phi_{f(i)}(\langle x,u\rangle)$ is constructed from $\phi_p(x,t)$ as follows: begin by running $\phi_p(x,t)$ for all $t \leq (u+i)$ for $(u+i)$ steps at most. If no such $\phi_p(x,t)$ converges in time $(u+i)$ or less, then dovetail the computations of $\phi_p(x,t)$ for all t, and find the first v where $\phi_p(x,v)$ converges. If w is the largest value where $\phi_p(x,w)$ converges in this computation and $\phi_p(x,w) = y$, then run $\phi_y(\langle x,u\rangle)$. If $\phi_y(\langle x,u\rangle)$ converges, where $\phi_y(\langle x,u\rangle) = z$ then output $z+1$.

By the assumption, for all x, $P_p(x)$ l-converges to the index of a total recursive function y, although the initial guesses may be of partial recursive functions. If $\phi_{f(i)}(\langle x,u\rangle)$ uses one of these guesses, it will never halt, at least on a finite number of initial values of u. But there will come a time s where for all $t \geq s$ the function $\phi_{f(t)}(\langle x,u\rangle)$ converges to a value for every u and is therefore total.

Choose j, $f(t) = j$ where ϕ_j is total. Since P_p is a complete enumeration, there will be an index y such that $P_p(y) = j$ in the limit, so after a finite value u, for all $t > u$ $\phi_p(\langle y,t\rangle) = j$ if it converges. Let $\phi_j(\langle y,t\rangle) = z$. This value z exists, since ϕ_j is total. But by the construction of $\phi_{f(t)} = \phi_j$ given above, $\phi_{f(t)}(\langle y,s\rangle) = \phi_j(\langle y,s\rangle) = \phi_j(\langle y,s\rangle) + 1$ a contradiction. So the Complete Enumeration of all Total Recursive Functions is not Computable in the Limit by a l-total P_p. □

We can strengthen this result by showing that no Complete Enumeration of even a single total recursive function is Computable in the Limit by any P_p, even if it is l-partial.

Theorem 5.3. *Given any total recursive function* TM_i*, the Complete Enumeration of all Total Functions* TM_j *equal to* TM_i *is not Computable in the Limit by any* P_p.

Proof. Assume the opposite: for an arbitrary TM_i, there is a P_p such that for all j, $TM_i = TM_j$ iff there is an x such that $P_p(x) = j$. Let $P_p(x)$ be computed by $\phi_p(x,t)$.

Given i and ϕ_p, define $TM_n(y)$ as follows. Run all $\phi_p(x,t)$ computations for $x \leq y$ and $t \leq y$ for up to $\Phi_p(x,t) \leq y$. If there is no case where $\phi_p(x,t) = n$ then $TM_n(y) = TM_i(y)$.

Otherwise, enumerate all cases $\langle x,t\rangle$ where $\phi_p(x,t) = n$ for $x \leq y$ and $t \leq y$ in time $\Phi_p(x,t) \leq y$. Let $y = \tau(m,q)$ and select the m^{th} case $\langle z,s\rangle$ where $\phi_p(z,s) = n$. This ensures that each such case gets chosen an infinite number of times during the

computations of all inputs y to $TM_n(y)$. Dovetail the computations of all $\phi_p(z, r)$ for $r > s$ until a value v is found where $\phi_p(z, v) = m$ and $m \neq n$. If none is found, then $TM_n(y)$ diverges. Otherwise, $TM_n(y) = TM_i(y)$.

If $P_p(y)$ never equals n for all y, then $TM_n = TM_i$. Then P_p cannot be a Complete Enumeration, since it missed TM_n.

If $P_p(y) = n$ for some y then there is a value s where $\phi_p(y, s) = n$. At this point $\phi_p(y, t)$ either diverges or $\phi_p(y, t) = n$ for all $t > s$. Let $TM_n(v)$ be a value v where this value $\phi_p(y, s) = n$ is the case selected in the computation of $TM_n(v)$. By the construction of TM_n, $TM_n(v)$ diverges, and is therefore not a total function.

Since this construction is true for any i and ϕ_p, the Complete Enumeration of all Total Functions TM_j equal to TM_i is not Computable in the Limit by any P_p. □

6. Conclusions

The class of properties that are Computable in the Limit are an interesting extension of the effectively computable functions. Although they are not computable in a finite time, they model our everyday notion of what learning and generalization are. The restricted subset of problems termed "Identification in the Limit" have been extensively covered in the Machine Learning and Computational Learning Theory fields. But the formal properties of the class itself, outside of its use as a model for learning, is itself interesting. This paper serves as a start for the exploration of this class.

It is obvious that the substitution of $\mu(x)$ for $\lambda(x)$ in the Normal Form Theorem can be further extended to classes where the computing function changes its mind an infinite number of times, and there is some predicate that defines a sort of limiting condition of that sequence. This would make it possible to extend Theorem 4.6 to the reals. This extension would of course be another superset, with characteristics all its own. How these levels of computation correspond to the Arithmetical Hierarchy is an open question.

7. Acknowledgements

I would like to acknowledge Patrick Boyle and David Buhanan for helpful discussions.

References

[1] L. Blum and M. Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28(2):125–155, 1975.
[2] E.M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.
[3] M. Li and P.M.B. Vitanyi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer Verlag, 1997.
[4] T. Ord. The many forms of hypercomputation. *Applied mathematics and computation*, 178(1):143–153, 2006.
[5] H. Rogers Jr. Theory of recursive functions and effective computability, 1967.
[6] R.J. Solomonoff. A formal theory of inductive inference. part 1. *Information and Control*, 7(1):1–22, 1964.
[7] R.J. Solomonoff. A formal theory of inductive inference. part 2. *Information and Control*, 7(2):224–254, 1964.

131 Wingate Drive, Hackettstown NJ, 07840
E-mail address: vandermude@acm.org
URL: www.vandermude.com