Lesson 2

Identification in the Limit

2.1 One reason defining learning is hard

A definition of a learning problem requires specifying the instances of the problem and specifying what counts as correct answers for these instances. This means thinking carefully about an interaction between three items: the learning targets, the learning algorithm, and the input to the learning algorithm, which can be thought of as the available evidence.

This is difficult because we have to confront the question "Which inputs is it reasonable to expect the learning algorithm to succeed on?" For example, if we are trying to identify a stringset S which is of infinite size but the evidence for S contains only a single string $s \in S$ then we may feel this places an unreasonable burden on the learning algorithm. What is at stake here was expressed by Charles Babbage:

> On two occasions I have been asked [by members of Parliament], "Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?" I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question. *as quoted in de la Higuera (2010, p. 391)*

It's unfair to expect a summation algorithm to succeed if the input is wrong. More generally, how do we define learning in such a way so that the input to the algorithm is not "wrong". What does it mean to have input of sufficient quality in learning? We want to only consider instances of the learning problem that are reasonable or fair. But nailing that down precisely is hard! In fact, what we will see is that this is an ongoing issue and there are many attempts to address it. The issue is a live one today.

2.2 Identification in the limit

Gold (1967) provided some influential definitions of learning. He called his approach **identification in the limit**. He provided not one, but several definitions, and he compared what kinds of stringsets were learnable in these paradigms.

No one I know knows what happened to Gold. He seems to have disappeared from academia in the 1980s.

Gold conceptualized learning as a never-ending process unfolding in time. Evidence is presented piece by piece in time to the learning algorithm. The learning algorithm outputs a program with each piece of evidence it receives based on its experience up to the present moment. As time goes on, the programs the learning algorithm outputs must be identical and they must exactly represent the target concept to be learned. In the case of learning formal languages, this means that given a data presentation for a formal language L, the learning algorithm must begin to consistently output a grammar G and that the language of G must equal L.

Let us explain the notation in the figure. The notation $t(n)$ means the evidence presented at time n. This notation suggests that evidence can be understood as a function with domain $\mathbb{N}$.

The notation t_n refers to the sequence of evidence up to the nth one. For example, t_3 means the finite sequence $t(1), t(2), t(3)$. In mathematics, angle brackets are sometimes used to denote sequences so some mathematicians would write this sequence as $\langle t(1), t(2), t(3) \rangle$.

Time t	1	2	3	4	...	n	...
Evidence at time t	$t(1)$	$t(2)$	$t(3)$	$t(4)$	...	$t(n)$	...
Input to Algorithm at time t	t_1	t_2	t_3	t_4	...	t_n	...
Output of Algorithm at time t	G_1	G_2	G_3	G_4	...	G_n	...

Figure 2.1: A schema of the Identification in the Limit learning paradigm

The notation G_n refers to the program output by the algorithm with input t_n. If A is the algorithm and n is any point in time, we can write $A(t_n) = G_n$.

There are two important ideas in this paradigm. First, a successful learning algorithm is one that converges over time to a correct generalization. At some time point n, the algorithm must output the same program and this program must solve the membership problem for S. This means the algorithm can make mistakes, *but only finitely many times*.

Second, which infinite sequences of evidence learners must succeed on? Which are the ones of sufficient quality? Gold defined required these sequences to be representative of the target stringsets. *Each* possible piece of evidence *occurs at some point* in the unfolding sequence of evidence. Lest we think this is too good to be true, recall that the input to the learner at any given point n in time is the *finite* sequence t_n, and that to succeed, it is only allowed to make finitely many mistakes.

2.2.1 Definition of identification in the limit from positive data

The box below precisely defines the paradigm when learning from *positive* data. Let us define the "evidence" when learning from positive data more precisely. A **positive presentation** of a stringset S is a function $t : \mathbb{N} \to S$ such that t is onto. Recall that a function f is onto provided for every element y in its co-domain there is some element x in its domain such that $f(x) = y$. Here, this means for every string $s \in S$, there is some $n \in \mathbb{N}$ such that $t(n) = s$.

Definition 1 (Identification in the limit from positive data)**.**

1 Algorithm A *identifies in the limit from positive data* a class of stringsets C provided
2 for all stringsets $S \in C$,
3 for all positive presentations t of S,
4 there is some number $n \in \mathbb{N}$ such that
5 for all $m > n$,
6 • the program output by A on t_m is the same as the the program output
7 by A on t_n, and
8 • the program output by A on t_m solves the membership problem for
9 S.

Here is breakdown of what these lines mean.

Line 1 Establishes the name of the relationship between an algorithm A and a collection of stringsets C provided the definition holds.

Line 2 The algorithm must succeed for all $S \in C$.
Line 3 The algorithm must succeed for all positive presentations t of S.
Line 4 It succeeds on t for S if there is a point in time n
Line 5 such that for all future points in time m,
Lines 6-7 the output of A converges to the same program, and
Lines 8-9 the output of A correctly solves the membership problem for S.
This paradigm is also called **learning from text.**

2.3 Examples: Learning from positive data

2.3.1 The Strictly k-Piecewise Stringsets

Here we present an algorithm and prove that it identifies the Strictly k-Piecewise (SP_k) stringsets in the limit from positive data. SP stringsets were proposed to model aspects of long-distance phonotactics Heinz (2010a), motivated on typological and learnability grounds. The learning scheme discussed here exemplifies more general ideas Heinz (2010b); Heinz *et al.* (2012).

The notion of *subsequence* is integral to SP stringsets. Informally, a string u is subsequence of string v if one is left with u after erasing zero or more letters in v. For example, ab is a subsequence of $cccccacccccccccbccccccc$. Formally, u is a subsequence of v ($u \sqsubseteq v$) provided there are strings $x_1, x_2, \ldots x_n$ and strings $y_0, y_1, \ldots y_n$ such that $u = x_1 x_2 \ldots x_n$ and $v = y_0 x_1 y_1 x_2 y_2 \ldots x_n y_n$. It is the y_i strings that erased in v to leave u.

A stringset S is Strictly Piecewise if and only if it is closed under subsequence. In other words, if $s \in S$ then every subsequence of s is also in S.

A theorem shows that every SP stringset S has a basis in a finite set of strings (Rogers *et al.*, 2010). These strings can be understood as *forbidden* subsequences. That is any string $s \in \Sigma^*$ containing any one of the forbidden subsequences is not in S. Conversely, any string s which does not contain any forbidden subsequence belongs to S.

The same theorem shows that a SP stringset S can be defined in terms of a finite set of *permissible* subsequences. Because the set is finite, there is a longest string in this set. Let its length be k. In this case, any $s \in \Sigma^*$ belongs to S if and only if every one of its subsequences of length k or less is permissible.

In other words we can define SP_k stringsets as follows. Let a grammar G be a finite subset of Σ^* and let k be the length of a longest string in G. Let $\texttt{subseq}_k(s) = \{u \mid u \sqsubseteq s, |u| \leq k\}$. The "language of the grammar" $L(G)$ is defined as the stringset $\{s \mid \texttt{subseq}_k(s) \subseteq G\}$. We are going to be interested in the collection of stringsets SP_k, defined as those stringsets generated from grammars G with a longest string k. Formally,

$$\text{SP}_k \stackrel{\text{def}}{=} \{S \mid G \subseteq \Sigma^{\leq k}, L(G) = S\} \; .$$

This is the collection C of learning targets.

For any presentation ϕ and time n, define k-SPIA (Strictly k-Piecewise Inference Algorithm) as follows

$$k\text{-SPIA}(t_n) = \begin{cases} \varnothing & \text{if } t = 0 \\ k\text{-SPIA}(t_{n-1}) \cup \texttt{subseq}_k(t(n)) & \text{otherwise} \end{cases}$$

Note that we are being a little sloppy here. Technically, the output of k-SPIA given some input sequence is a set of subsequences G, not a program. What we really mean with the above is that k-SPIA outputs a program which uses G to solve the membership problem for $L(G) = \{w \mid \texttt{subseq}_k(w) \subseteq G\}$. This program looks something like this.

1. INPUT: any word w.
2. If $\texttt{subseq}_k(w) \subseteq G$ then OUTPUT Yes, otherwise OUTPUT No.

All k-SPIA does is update this program simply by updating the contents of G.

Theorem 1. *For each k, k-SPIA identifies in the limit from positive data the collection of stringsets SP_k.*

Proof. Consider any $k \in \mathbb{N}$. Consider any $S \in \text{SP}_k$. Consider any positive presentation t for S. It is sufficient to show there exists a point in time n_ℓ such that for all $m \geq n_\ell$ the following holds:

1. k-SPIA$(t_m) = k$-SPIA(t_{n_ℓ}) (convergence), and
2. k-SPIA(t_m) is a program that solves the membership problem for S.

Since $S \in \text{SP}_k$, there is a finite set $G \subseteq \Sigma^{\leq k}$ such that $S = L(G)$.

Consider any subsequence $g \in G$. Since $g \in G$ there is some word $w \in S$ which contains g as a k-subsequence. Since G is finite, there are finitely many such w, one for each g in G. Because t is a positive presentation for S, there is a time t where each of these w occurs. For each w let n_w be the first occurrence of w in t. Let n_ℓ denote the latest time point of all of these time points n_w.

Consider any $m \geq n_\ell$. The claim is that k-SPIA$(t_m) = k$-SPIA$(t_{n_\ell}) = G$. For each g in G, a word containing g as a subsequence occurs at or earlier than n_ℓ and so $g \in k$-SPIA(t_m). Since g was arbitrary in G, $G \subseteq k$-SPIA(t_m).

Similarly, for each $g \in k$-SPIA$(_m)$, there was some word w in t such that w contains g as a subsequence. Since t is a positive presentation for S, w is in S. Since w belongs to S, $\texttt{subseq}_k(w) \subseteq G$ and so g belongs to G. Since g was arbitrary in k-SPIA$(_m)$ it follows that k-SPIA$(_m) \subseteq G$.

It follows k-SPIA$(_m) = G$.

Since m was arbitrarily larger than n_ℓ we have both convergence and correctness.

Since t was arbitrary for S, S arbitrary in SP_k and k arbitrary, the proof is concluded. □

2.3.2 The Strictly k-Local Stringsets

Here we present an algorithm and prove that it identifies the Strictly k-Local (SL_k) stringsets in the limit from positive data. The first proof of this result was presented by Garcia *et al.* (1990), though the Markovian principles underlying this result were understood in a statistical context much earlier. The learning scheme discussed there exemplifies more general ideas (Heinz, 2010b; Heinz *et al.*, 2012).

The notion of *substring* is integral to SL stringsets. Formally, a string u is substring of string v ($u \trianglelefteq v$) provided there are strings $x, y \in \Sigma^*$ and $v = xuy$. Another term for substring is *factor*. So we also say that u is a factor of v. If u is of length k then we say u is a k-factor of v.

A stringset S is Strictly k-Local if and only if there is a number k such that for all strings $u_1, v_1, u_2, v_2, x \in \Sigma^*$ such that if $|x| = k$ and $u_1xv_1, u_2xv_2 \in S$ then $u_1xv_2 \in S$. We say S is closed under suffix substitution (Rogers and Pullum, 2011).

A theorem shows that every SL_k stringset S has a basis in a finite set of strings (Rogers and Pullum, 2011). These strings can be understood as *forbidden* substrings. Informally, this means

any string s containing any one of the forbidden substrings is not in S. Conversely, any string s which does not contain any forbidden substring belongs to S.

The same theorem shows that a SL stringset S can be defined in terms of a finite set of *permissible* substrings. In this case, s belongs to S if and only if every one of its k-factors is permissible.

We formalize the above notions by first defining a function the $\texttt{factor}_k$, which extracts the substrings of length k present in a string, or those present in a set of strings. If a string s is of length less than k then $\texttt{factor}_k$ just returns s.

Formally, let $\texttt{factor}_k(s)$ equal $\{u \mid u \trianglelefteq s, |u| = k\}$ whenever $k \leq |s|$ and let $\texttt{factor}_k(s) = \{s\}$ whenever $|s| < k$. We expand the domain of this function to include sets of strings as follows: $\texttt{factor}_k(S) = \bigcup_{s \in S} \texttt{factor}_k(s)$.

To formally define SL_k grammars, we introduce the symbols $\rtimes$ and $\ltimes$, which denote left and right word boundaries, respectively. These symbols are introduced because we also want to be able to forbid specific strings at the beginning and ends of words, and traditionally strictly local stringsets were defined to make such distinctions (McNaughton and Papert, 1971). Then let a grammar G be a finite subset of $\texttt{factor}_k(\{\rtimes\}\Sigma^*\{\ltimes\})$.

The "language of the grammar" $L(G)$ is defined as the stringset $\{s \mid \texttt{factor}_k(\rtimes s \ltimes) \subseteq G\}$. We are going to be interested in the collection of stringsets SL_k, defined as those stringsets generated from grammars G with a longest string k. Formally,

$$\text{SL}_k \stackrel{\text{def}}{=} \{S \mid G \subseteq \texttt{factor}_k(\{\rtimes\}\Sigma^*\{\ltimes\}), L(G) = S\} \ .$$

This is the collection C of learning targets.

For all $S \in \text{SL}_k$, for any presentation ϕ and time n, define k-SLIA (Strictly k-Local Inference Algorithm) as follows

$$k\text{-SLIA}(t_n) = \begin{cases} \varnothing & \text{if } t = 0 \\ k\text{-SLIA}(t_{n-1}) \cup \texttt{factor}_k(\rtimes t(n) \ltimes) & \text{otherwise} \end{cases}$$

Exercise 1. *Prove algorithm k-SLIA identifies in the limit from positive data the collection of stringsets SL_k.*

Note that we are being a little sloppy here. Technically, the output of k-SLIA given some input sequence is a set of factors G, not a program. What we really mean with the above is that k-SLIA outputs a program which uses G to solve the membership problem for $L(G) = \{w \mid \texttt{factor}_k(\rtimes w \ltimes) \subseteq G\}$. This program looks something like this.

1. INPUT: any word w.
2. If $\texttt{factor}_k(\rtimes w \ltimes) \subseteq G$ then OUTPUT Yes, otherwise OUTPUT No.

All k-SLIA does is update this program simply by updating the contents of G.

Theorem 2. *For each k, k-SLIA identifies in the limit from positive data the collection of stringsets SL_k.*

Proof. Consider any $k \in \mathbb{N}$. Consider any $S \in \text{SL}_k$. Consider any positive presentation t for S. It is sufficient to show there exists a point in time n_ℓ such that for all $m \geq n_\ell$ the following holds:

1. $k\text{-SLIA}(t_m) = k\text{-SLIA}(t_{n_\ell})$ (convergence), and
2. $k\text{-SLIA}(t_m)$ is a program that solves the membership problem for S.

Since $S \in \text{SL}_k$, there is a finite set $G \subseteq \Sigma^{\leq k}$ such that $S = L(G)$.

Consider any factor $g \in G$. Since $g \in G$ there is some word $w \in S$ which contains g as a k-factor. Since G is finite, there are finitely many such w, one for each g in G. Because t is a positive presentation for S, there is a time t where each of these w occurs. For each such w let n_w be the first occurrence of w in t. Let n_ℓ denote the latest time point of all of these time points n_w.

Consider any $m \geq n_\ell$. The claim is that k-SLIA(t_m) = k-SLIA$(t_{n_\ell}) = G$. For each g in G, a word containing g as a factor occurs at or earlier than n_ℓ and so $g \in k$-SLIA(t_m). Since g was arbitrary in G, $G \subseteq k$-SLIA(t_m).

Similarly, for each $g \in k$-SLIA(t_m), there was some word w in t such that w contains g as a factor. Since t is a positive presentation for S, w is in S. Since w belongs to S, $\texttt{factor}_k(w) \subseteq G$ and so g belongs to G. Since g was arbitrary in k-SLIA(t_m) it follows that k-SLIA$(t_m) \subseteq G$.

It follows k-SLIA$(_m) = G$.

Since m was arbitrarily larger than n_ℓ we have both convergence and correctness.

Since t was arbitrary for S, S arbitrary in SL_k and k arbitrary, the proof is concluded. □

2.4 Results

In this chapter review general results in the identification in the limit paradigm. We begin with some theorems for learning from positive data.

We have already seen that the following classes of languages are identifiable in the limit from positive data.

1. For each $k \in \mathbb{N}$, SP_k
2. For each $k \in \mathbb{N}$, SL_k
3. BAR-X = $\{\bar{x} \mid x \in \Sigma^*\}$. Recall $\bar{x} \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid w \neq x\}$.

The following classes of stringsets are fundamental ones in formal language theory so it makes sense to be curious about their learnability.

1. The class of finite stringsets (FIN).
2. The class of regular stringsets (REG).
3. The class of context-free stringsets (CF).
4. The class of context-sensitive stringsets (CS).

These are in the following relationship: FIN $\subsetneq$ REG $\subsetneq$ CF $\subsetneq$ CF.

2.5 Identification in the limit from positive data

Theorem 3. *FIN is identifiable in the limit from positive data.*

Exercise 2. *Prove this theorem. (Hint: FIN can be learned with string extension learning.)*

Any class which includes every finite language and at least one more is a *superfinite* class of languages.

Theorem 4. *No superfinite class of stringsets is identifiable in the limit from positive data.*

There are different ways to prove this theorem. Here is one based on (de la Higuera, 2010, p. 151).

sketch. Consider any superfinite class of languages C. By definition C includes all finite languages and at least one infinite language L_∞.

Let $x_1, x_2, \ldots$ be the infinitely many words of L_∞.

Let $L_1 = \{x_1\}$, $L_2 = L_1 \cup \{x_2\}$, $L_3 = L_2 \cup \{x_3\}$, and so on. So $L_k = L_{k-1} \cup \{x_k\}$. For each $k \in \mathbb{N}$, $L_k \in C$ since L_k is finite.

For the sake of contradiction, assume there is an algorithm A that identifies C in the limit from positive data. We will show there is a presentation for L_∞ for which A fails to converge.

Pick a presentation t_1 for L_1. Since A identifies L_1 in the limit, there is a convergence point i_1 such that A outputs a grammar for L_1 on $t_1[i_1]$. Let t_2 be some presentation of L_2 such that for all $j < i_1$, $t_2(j) = t_1(j)$. In other words t_2 is the same as t_1 up to point i_i. Since A identifies L_2 in the limit, there is a convergence point $i_2 > i_1$ such that A outputs a grammar for L_2 on $t_2[i_2]$. More generally, let t_k be some presentation of L_k such that for all $j < i_{k-1}$, $t_k(j) = t_{k-1}(j)$ and $t_k(i_{k-1} + 1) = x_k$. Since A identifies L_k in the limit, there is a convergence point i_k such that A outputs a grammar for L_k on $t_k[i_k]$.

In this manner we construct a presentation t_∞ for L_∞ as follows. For any $i \in \mathbb{N}$, there exists $j, j+1$ such that $i_j < i \leq i_{j+1}$. Let k equal $j + 1$. Then $t_\infty(i)$ equals $t_k(i)$.

How does A behave on t_∞? It does not converge. This is because for all $k \in \mathbb{N}$, at time point i_k, A will output a program for L_k. So it never converges to a grammar for L_∞ even though t_∞ is a positive presentation for L_∞. □

Gold explains the idea behind his result this way.

> It is of great interest to find why information presentation by text is so weak and under what circumstances it becomes stronger. Therefore, it is worthwhile to understand the method used in Theorems I.8 and I.9 to prove that any class of languages containing all finite languages and at least one infinite language is not identifiable in the limit from a text in five out of six of the models using text.
>
> The basic idea is proof by contradiction. Consider any proposed guessing algorithm. It must identify any finite language correctly after a finite amount of text. This makes it possible to construct a text for the infinite language which will fool the learner into making a wrong guess an infinite number of times as follows. The text ranges over successively larger, finite subsets of the infinite language. At each stage it repeats the elements of the current subset long enough to fool the learner. Thus, the method of proof of the negative results concerning text depends on the possibility of there being a huge amount of repetition in the text. Perhaps this can be prevented by some reasonable probabilistic assumption concerning the generation of the text. In this case one would only require identification in the limit with probability one, rather than for every allowed text.
>
> I have been asked, "If information presentation is by means of text, why not guess the unknown language to be the simplest one which accepts the text available?" This is identification by enumeration. It is instructive to see why it will not work for most interesting classes of languages: The universal language (if it is in the class) will have some finite complexity. If the unknown language is more complex, then the guessing procedure being considered will always guess wrong, since the universal language is consistent with any finite text. This follows from the fact that, if L is the unknown

> language and if L' $\supset$ L, then L' is consistent with any finite segment of any text for L. The problem with text is that, if you guess too large a language, the text will never tell you that you are wrong.

It immediately follows that the class of regular, context-free, context-sensitive, and computably enumerable classes of stringsets are not identifiable in the limit from positive data.

Furthermore, for every finite stringset S, there is some k such that S is Strictly k-Local. Thus FIN $\subsetneq$ SL. Hence neither SL nor LT nor LTT nor TSL is identifiable in the limit from positive data.

Theorem 5 (Angluin 1980). *A class C is identifiable in the limit from positive data iff for each $S \in C$ there is a finite set $D \subseteq S$ such that for all $S' \in C$ such that $D \subseteq S'$ it holds that $S' \not\subseteq S$.*

Pictorially, Figure 2.2 is the situation that cannot obtain.

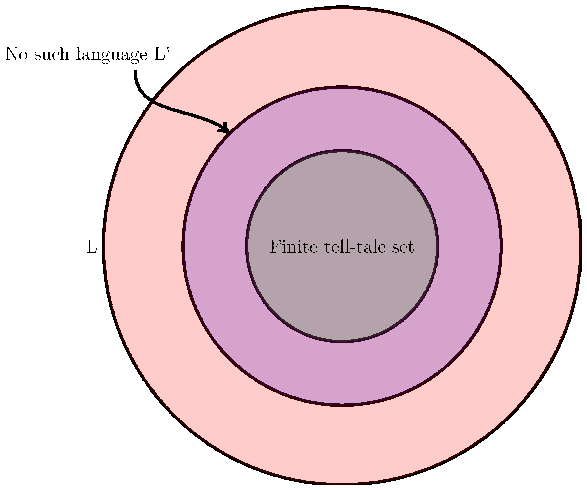

Figure 2.2: No such L' in every class identifiable in the limit from positive data!

Corollary 1. *Every finite class of languages is identifiable in the limit from positive data.*

Gold's theorems and Angluin's theorems above are the basis for the so-called "subset problem" in the linguistics literature on learning (Wexler and Culicover, 1980; Berwick, 1985).

The proof of the previous theorem ought to be presented.

2.6 Identification in the limit from positive and negative data

A **positive and negative presentation** of a stringset S provides example strings not in S in addition to example strings in S. This can be formalized using the *characteristic function* of S. Every set S has a characteristic function with domain Σ^* defined as follows.

$$f_S(s) = \begin{cases} 1 & \text{iff } s \in S \\ 0 & \text{otherwise} \end{cases}$$

Characteristic functions are total functions, which means defined for all $s \in \Sigma^*$. Also recall, that we write $(x, y) \in f$ whenever $f(x) = y$. So we can think of f_S as a set of points where $(s, 0)$ means $s \notin S$ and $(s, 1)$ means $s \in S$.

Then a **positive and negative presentation** of a stringset S is a function $\varphi : \mathbb{N} \to f_S$ such that φ is onto. Here, this means for every string $s \in \Sigma^*$, there is some $n \in \mathbb{N}$ such that $\varphi(n) = (s, f_s(s))$.

Definition 2 (Identification in the limit from positive and negative data)**.**

Algorithm A *identifies in the limit from positive and negative data* a class of stringsets C provided

for all stringsets $S \in C$,
for all positive and negative presentations t of S,
there is some number $n \in \mathbb{N}$ such that
for all $m > n$,
- the program output by A on t_m is the same as the the program output by A on t_n, and
- the program output by A on t_m solves the membership problem for S.

The only difference between the definition above and the one in Definition 1 is in line 3. This paradigm is also called **learning from an informant.**

Theorem 6. *The computable class of languages is identifiable in the limit from positive and negative data.*

sketch. The algorithm proceeds by enumeration over programs. Since programs are strings, we can essentially use the enumeration for strings we used before.

The learning algorithm finds the first program in the enumeration that successfully classifies all of the data it has observed so far in t.

How does it do this? Well, it looks at the first program in the enumeration and submits to it each data point t_i. If the first program fails to compute anything, or classifies any one of the data points incorrectly, the learning algorithm moves to the next program and checks again. This repeats. Eventually, it must find a program which classifies all of the observed data points correctly. At this point, it outputs this program.

How do we know this algorithm converges to a correct program for S? Well, there is a program for S in the enumeration. There may be more than one such program so let P be the first program in the enumeration for S. Once the learning algorithm reaches P it will output P since P will classify all data points from t correctly because t is a positive and negative presentation of S.

How do we know the learning algorithm will eventually output P on any positive and negative data presentation t for S? Consider any program P' prior to P in the enumeration. Since P is the first program in the enumeration for S, P' is not a program for S. It follows that there is some $w \in S$, or $w \notin S$ that P' misclassifies. It follows that there is some point in time i such that $t(i) = (w, x)$ with $x \in \{0, 1\}$. At this point the learning algorithm will conclude that P'

does not classify everything it has seen in $t[i]$ correctly, and will move to the next program in the enumeration. Since P' was arbitrary, it follows that the algorithm will eventually reach and output program P. □

2.7 Identification in the limit from primitive recursive texts

Recall that algorithms have to succeed for *any* text or from *any* informant. As we discuss later, this has been one source of criticism of Gold's learning paradigm.

Let's consider texts for the moment. That is, let's consider positive presentations for some stringset S which has at least two strings in it. How many positive presentations are there? It should be easy to see that there are infinitely many. If u, v are distinct words in S then a text could start either u, v or u, u, v or u, u, u, v or u, u, u, u, v and so on. In fact, there are *uncountably many* presentations for S. This can be shown by the same diaganolization argument that we used earlier to show that there are uncountably many subsets of Σ^*.

We can also ask how many of these presentations are computable? Provided there are at least two strings in S, the answer is only countably many.

This situation is exactly analogous to the number of real numbers between 0 and 1, inclusive, and the number of computable real numbers between 0 and 1, inclusive.

In other words, *most* of the presentations that the Gold paradigm is required to succeed on are *uncomputable*. Some have argued this is not reasonable.

Regardless of whether it is or not, we may be interested in what changes if we change the definition of learning to only require success on computable texts.

A particularly strong form of computability is computability via *primitive recursion*. This is weaker than Turing-machine computable. For example, Turing machines are not guaranteed to halt on input, but primitive recursive programs are guaranteed to halt. See Rogers (1967) for details on primitive recursion.

Definition 3 (Identification in the limit from primitive recursive texts)**.**

1 Algorithm A *identifies in the limit from positive data* a class of stringsets C provided
2 for all stringsets $S \in C$,
3 for all positive presentations t of S definable with primitive recursive functions,
4 there is some number $n \in \mathbb{N}$ such that
5 for all $m > n$,
6 • the program output by A on t_m is the same as the the program output
7 by A on t_n, and
8 • the program output by A on t_m solves the membership problem for
9 S.

The only difference between the definition above and the ones in Definitions 1 and 2 is in line 3.

Theorem 7. *The recursive (computable) class of languages is identifiable in the limit from primitive recursive texts.*

I'm not able at present to explain in detail this proof, so it is omitted. I think the basic ideas are that (1) primitive recursive texts are enumerable and (2) it is possible to translate a primitive recursive text into a grammar which recognizes a language equal to the content of the text (set of strings in the text). So an algorithm can identify by enumeration the primitive recursive text and thus the language the text is from.

Note there are distinctions between primitive recursive, recursive, and Turing computable. Another result of Gold's is that identification in the limit of superfinite language classes from positive data definable with *recursive* functions is still impossible. These are worth studying in more detail.

2.8 Gold's interpretation of these results

From (Heinz, 2016):

Gold (1967:453-454) provides three ways to interpret his three main results:

> 1. The class of natural languages is much smaller than one would expect from our present models of syntax. That is, even if English is context-sensitive, it is not true that any context-sensitive language can occur naturally...In particular the results on [identification in the limit from positive data] imply the following: The class of possible natural languages, if it contains languages of infinite cardinality, cannot contain all languages of finite cardinality.
> 2. The child receives negative instances by being corrected in a way that we do not recognize...
> 3. There is an a priori restriction on the class of texts [presentations of data; i.e. infinite sequences of experience] which can occur...

The first possibility follows directly from the fact that no superfinite class of languages is identifiable in the limit from positive data. The second and third possibilities follow from Gold's other results on *identification in the limit from positive and negative data* and on *identification in the limit from positive primitive recursive data* ...

Each of these research directions can be fruitful, if honestly pursued. For the case of language acquisition, Gold's three suggestions can be investigated empirically. We ought to ask

1. What evidence exists that possible natural language patterns form subclasses of major regions of the Chomsky Hierarchy?
2. What evidence exists that children receive positive and negative evidence in some, perhaps implicit, form?
3. What evidence exists that each stream of experience each child is exposed to is guaranteed to be generated by a fixed, computable process (i.e. computable probability distribution or primitive recursion function)? More generally, what evidence exists that the data presentations are a priori limited?

My contention is that we have plenty of evidence with respect to question (1), some evidence with respect to (2), and virtually no evidence with respect to (3).

Finally, Gold concludes his paper this way.

> Concerning inductive inference, philosophers often occupy themselves with the following type of question: Suppose we are given a body of information and a set of

> possible conclusions, from which we are to choose one. Some of the conclusions are eliminated by the information. The question is, of the conclusions which are consistent with the in- formation, which is "correct"?
>
> If some sort of probability distribution is imposed on the set of conclusions, then the problem is meaningful. But if no basis for choosing between the consistent conclusions is postulated a priori, then inductive inference can do no more than state the set of consistent conclusions.
>
> The difficulty with the inductive inference problem, when it is stated this way, is that it asks, "What is the correct guess at a specific time with a fixed amount of information?" There is no basis for choosing between possible guesses at a specific time. However, it is interesting to study a guessing strategy. Now one can investigate the limiting behavior of the guesses as successively larger bodies of information are considered. This report is an example of such a study. Namely, in interesting identification problems, a learner cannot help but make errors due to incomplete knowledge. But, using an "identification in the limit" guessing rule, a learner can guarantee that he will be wrong only a finite number of times.

2.9 Criticisms

1. Identification in the limit from positive data is too hard. The texts can be adversarial and exact convergence is unreasonable.
2. Identification in the limit doesn't address time or resource complexity of learning, which is also unreasonable.

The first point, particularly the adversality of the texts, is argued by Clark and Lappin (2011). Of course the texts are also complete in the sense that every string in the target language eventually occurs. So while there are adversarial texts, the adversity is somewhat constrained.

The second point is about feasibility. Learning by enumeration is very, very far from efficient. So even if every finite class of languages is identifiable in the limit from positive data, large finite classes may not be efficiently learnable because learning by enumeration is awfully slow! Similarly, Even if the recursive class is identifiable in the limit from primitive recursive text, it is not efficiently learnable. So we need some way to identify feasibly learnable subclasses. On the other hand, as defined identification in the limit paradigms can be understood about what is even possible; that is, whether any algorithm even exists which can exhibit the desired behavior.

Much research since Gold has aimed to incorporate feasibility into learning. The Probably Approximately Correct learning model is one influential example (Valiant, 1984; Anthony and Biggs, 1992; Kearns and Vazirani, 1994).

Many researchers advocate a learning setting where the aim is not to learn categorical stringsets but to learn probability distributions over them ("stochastic stringsets.") We will talk about this next.

The most repeated refrain ever in cognitive science, computational linguistics about the theory of learning languages is this: "Gold (1967) showed that context-free grammars are not learnable but Horning (1969) showed that probabilistic context-free grammars are." There is so much confusion about this, I wrote about it: (Heinz, 2016).

Bibliography

Angluin, Dana. 1980. Inductive inference of formal languages from positive data. *Information Control* 45:117–135.

Anthony, M., and N. Biggs. 1992. *Computational Learning Theory*. Cambridge University Press.

Berwick, Robert. 1985. *The acquisition of syntactic knowledge*. Cambridge, MA: MIT Press.

Clark, Alexander, and Shalom Lappin. 2011. *Linguistic Nativism and the Poverty of the Stimulus*. Wiley-Blackwell.

Garcia, Pedro, Enrique Vidal, and José Oncina. 1990. Learning locally testable languages in the strict sense. In *Proceedings of the Workshop on Algorithmic Learning Theory*, 325–338.

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

Heinz, Jeffrey. 2010a. Learning long-distance phonotactics. *Linguistic Inquiry* 41:623–661.

Heinz, Jeffrey. 2010b. String extension learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 897–906. Uppsala, Sweden: Association for Computational Linguistics.

Heinz, Jeffrey. 2016. Computational theories of learning and developmental psycholinguistics. In *The Oxford Handbook of Developmental Linguistics*, edited by Jeffrey Lidz, William Synder, and Joe Pater, chap. 27, 633–663. Oxford, UK: Oxford University Press.

Heinz, Jeffrey, Anna Kasprzik, and Timo Kötzing. 2012. Learning with lattice-structured hypothesis spaces. *Theoretical Computer Science* 457:111–127.

de la Higuera, Colin. 2010. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press.

Horning, J. J. 1969. A study of grammatical inference. Doctoral dissertation, Stanford University.

Kearns, Michael, and Umesh Vazirani. 1994. *An Introduction to Computational Learning Theory*. MIT Press.

McNaughton, Robert, and Seymour Papert. 1971. *Counter-Free Automata*. MIT Press.

Rogers, Hartley. 1967. *Theory of Recursive Functions and Effective Computability*. McGraw Hill Book Company.

Rogers, James, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome, and Sean Wibel. 2010. On languages piecewise testable in the strict sense. In *The Mathematics of Language*, edited by Christian Ebert, Gerhard Jäger, and Jens Michaelis, vol. 6149 of *Lecture Notes in Artifical Intelligence*, 255–265. Springer.

Rogers, James, and Geoffrey Pullum. 2011. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information* 20:329–342.

Valiant, L.G. 1984. A theory of the learnable. *Communications of the ACM* 27:1134–1142.

Wexler, Kenneth, and Peter Culicover. 1980. *Formal Principles of Language Acquisition*. MIT Press.