Chapter 1

Definitions of Learning

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.

1.1 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 must solve the membership problem for the target stringset.

Time t	1	2	3	4	...	n	...
Evidence at time t	e(1)	e(2)	e(3)	e(4)	...	e(n)	...
Input to Algorithm at time t	e$\langle 1 \rangle$	e$\langle 2 \rangle$	e$\langle 3 \rangle$	e$\langle 4 \rangle$	...	e$\langle n \rangle$	...
Output of Algorithm at time t	G(1)	G(2)	G(3)	G(4)	...	G(n)	...

Figure 1.1: A schema of the Identification in the Limit learning paradigm

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

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

The notation "G(n)" refers to the program output by the algorithm with input e$\langle n \rangle$. If A is the algorithm, and we wish to use functional notation so that $A(i) = o$ means "on input i, algorithm A outputs o" then G(n)=A(e$\langle n \rangle$).

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 e$\langle n \rangle$, and that to succeed, it is only allowed to make finitely many mistakes.

1.1.1 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 $\varphi : \mathbb{N} \to S$ such that φ 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 $\varphi(n) = s$.

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 φ 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 $\varphi\langle m \rangle$ is the same as the the program
7 output by A on $\varphi\langle n \rangle$, and
8 • the program output by A on $\varphi\langle m \rangle$ solves the membership problem
9 for 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 φ of S.

Line 4 It succeeds on φ 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.**

Example 1. 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, $u = \sigma_1\sigma_2\cdots\sigma_n$ is a subsequence of v ($u \sqsubseteq v$) if $u \in \Sigma^*\sigma_1\Sigma^*\sigma_2\Sigma^*\cdots\Sigma^*\sigma_n\Sigma^*$.

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 $\mathtt{subseq}_k(s) = \{u \mid u \sqsubseteq s, |u| \leq k\}$. The "language of the grammar" $L(G)$ is defined as the stringset $\{s \mid \mathtt{subseq}_k(s) \subseteq G\}$. 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,

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

This is the collection C of learning targets.

For all $S \in \mathrm{SP}_k$, all presentations φ of S, and all time points $t \in \mathbb{N}$ define A as follows:

$$A\big(\varphi\langle t\rangle\big) = \begin{cases} \mathtt{subseq}_k(\varphi(t)) & \text{if } t = 1 \\ A(\varphi\langle t-1\rangle) \cup \mathtt{subseq}_k(\varphi(t)) & \text{otherwise} \end{cases}$$

One can prove that algorithm A identifies in the limit from positive data the collection of stringsets SP_k.

Exercise 1. Prove algorithm A identifies in the limit from positive data the collection of stringsets SP_k.

1.1.2 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)**.**

10 Algorithm A *identifies in the limit from positive and negative data* a class of stringsets
11 C provided
12 for all stringsets $S \in C$,
13 for all positive and negative presentations φ of S,
14 there is some number $n \in \mathbb{N}$ such that
15 for all $m > n$,
16 • the program output by A on $\varphi\langle m \rangle$ is the same as the the program
17 output by A on $\varphi\langle n \rangle$, and
18 • the program output by A on $\varphi\langle m \rangle$ solves the membership problem
19 for S.

The only difference in the definitions is in line 3. This paradigm is also called **learning from an informant.**

1.1.3 Variations on a theme

Exercise 2. Suppose we want to learn a transformation from strings to strings? In other words a relation from Σ^* to Δ^*? How could the definitions above be changed?

Exercise 3. Suppose we want to learn a probability distribution over Σ^*. How could the definitions above be changed?

Bibliography

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, 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.

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.