Learners Based on Transducers

Sanjay Jain[1(✉)], Shao Ning Kuek[2], Eric Martin[3], and Frank Stephan[1,2]

[1] Department of Computer Science, National University of Singapore, 13 Computing Drive, COM1, Singapore 117417, Republic of Singapore
{sanjay,fstephan}@comp.nus.edu.sg

[2] Department of Mathematics, National University of Singapore, 10 Lower Kent Ridge Road, S17, Singapore 119076, Republic of Singapore
shaoning@u.nus.edu

[3] School of Computer Science and Engineering, The University of New South Wales, Sydney, NSW 2052, Australia
eric.martin@unsw.edu.au

Abstract. As data come out one by one from an infinite stream, automatic learners maintain some string as long term memory, and update it at every new datum (example) they process. Transduced learners are generalization of automatic learners. Both kind of learners are evaluated with respect to the space they consume for learning. For automatic learners, it is unknown whether at any point, the size of the long term memory can be bounded by the length of the longest datum that has been received so far. Here it is shown that, even when restricting learning to automatic families, there is a hierarchy of classes that can be learnt with memory $O(n^k)$, and all automatic families which are learnable in principle can be learnt by a transduced learner using exponential sized memory.

Keywords: Automata and logic · Inductive inference · Transducers

1 Introduction

Gold [11] introduced the model of learning in the limit from positive data. Subsequent research in inductive inference [1,3,6,15,19,22,23] studied also variations on this model. The basic features of Gold's model are the following. Let $L \subseteq \Sigma^*$ be a language, where Σ is a finite alphabet. The learner gets as input a *text for* L, that is, a sequence of strings x_0, x_1, ... that contains all members but no non-member of L. As output, the learner conjectures a sequence of indices e_0, e_1, ... as its hypotheses on what the input language might be. The hypotheses are taken from some hypothesis space $\{H_e : e \in I\}$, where I is the set of possible indices and every possible learning task equals some H_e. If the sequence

S. Jain and F. Stephan are supported in part by the Singapore Ministry of Education Academic Research Fund grants R146-000-181-112 and MOE2016-T2-1-019 / R146-000-234-112. Furthermore, S. Jain is supported in part by NUS grant C252-000-087-001. F. Stephan did part of the work while on sabbatical leave to UNSW Sydney.

S. T. Klein et al. (Eds.): LATA 2018, LNCS 10792, pp. 169–181, 2018.
https://doi.org/10.1007/978-3-319-77313-1_13

of hypotheses converges to an index e for the language L (that is, $H_e = L$), then the learner is said to have learnt the input language from the text. The learner learns a language L if it learns it from all texts for L. It learns a class $\mathcal{L}$ of languages if it learns all languages in $\mathcal{L}$. To measure the complexity of a learner, it is convenient to consider the learner as operating in cycles: it starts with hypothesis e_0 and in the n-th cycle, it gets the datum x_n and conjectures the hypothesis e_{n+1}. Freivalds et al. [10] and Kinber and Stephan [18] imposed the condition that between two cycles, the learner remembers only part of its previous inputs and works via some (long term) memory, which can be restricted. Thus, the complexity of learners can be measured in terms of two parameters: (a) the computational complexity of mapping the old memory and input datum to the new memory and hypothesis and (b) the length of the memory as a function of the length of the longest example seen so far.

Fundamental choices for (a) are: recursive learners (the mapping can be computed by a Turing machine), transduced learners (the mapping can be computed by a finite transducer) and automatic learners (the mapping is an automatic function). Pitt [23] showed that many complexity-theoretic restrictions—like requiring that the update time in each cycle be carried out in time polynomial in the sum of the lengths of all inputs seen so far—do not give a real restriction for most learning criteria from classical inductive inference. Automatic learners are more severely restricted and offer an interesting object of study [5,13]. In particular, it is natural to investigate the target classes that are represented in an automata-theoretic framework, namely the automatic families [14]. These offer a representation that an automatic learner can handle easily. It is also natural to impose that hypothesis spaces be themselves automatic families containing the class to be learnt. It turns out that certain such families are learnable by a recursive learner, but not by an automatic learner. The inability to memorise all past data is a major weakness of automatic learners and is exploited by many non-learnability proofs. Still, as shown by Jain et al. [13], w.r.t. the learnability of automatic families from fat text (with infinitely many occurrences of each datum), automatic and recursive learners have the same power; their memory can even be restricted to the length of the longest datum seen so far, the so-called *word length memory limitation*.

The current work studies transduced learners which are a generalisation of automatic learners. For transduced learners, the update mapping of the learner is computed by a non-deterministic transducer which on all accepting runs, produces the same outputs for the same inputs. Both inputs (old memory and current datum) are read independently, and both outputs (new memory and hypothesis) are written independently. This independence makes transduced learners more powerful than automatic ones.

For (b), Freivalds et al. [10] imposed that learners operate in cycles and only remember, from one cycle to the next, information recorded in a (long term) memory, with restrictions on its length. For automatic and transduced learners, the memory is a string over a fixed alphabet, that may depend on the learner, whose size is measured by the length of the string [10,13,18]. In the subfield of

automatic learning, this way of restricting the memory led to fruitful findings, still leaving one major question open: is it truly restrictive to bound the memory by the length of the longest datum seen so far? A positive answer will be provided for transduced learners. Moreover, there is a learning hierarchy based on the memory sizes as a function of the length of the longest datum seen so far. In particular, polynomials and exponential functions of various degrees and exponents, respectively, are the most prominent memory bounds.

2 Preliminaries

Let $\mathbb{N}$ denote the set of natural numbers $\{0, 1, \ldots\}$. Let Σ denote a finite alphabet (set of symbols), ε the empty string, and Σ^* the set of all strings (words) over Σ. A language is a subset of Σ^* for some finite alphabet Σ. Concatenation of strings u and v is denoted by $u \cdot v$, or just uv when the context makes it clear. A string u of length n can be considered as a function from $\{0, 1, \ldots, n-1\}$ to Σ, with $u(i)$ the $(i+1)$-th symbol in u. Length lexicographic order between strings is defined as follows: $u <_{ll} v$ if either $|u| < |v|$ or $|u| = |v|$ and u is lexicographically before v w.r.t. some underlying ordering of Σ.

2.1 Automatic Relations and Functions

A relation $R = \{(u_1, \ldots, u_n) : u_i \in \Sigma^*\}$ is said to be *automatic* iff it is recognised by a finite automaton with n inputs which reads all inputs at the same speed (one symbol per input and cycle with a special symbol # when the corresponding input is exhausted) [4,6,12,16,17,24]. A function f with m inputs and n outputs is said to be automatic iff the corresponding relation, that is, $R = \{(u_1, \ldots, u_m, v_1, \ldots, v_n) : f(u_1, \ldots, u_m) = (v_1, \ldots, v_n)\}$, is automatic. A class of languages $\{L_e : e \in I\}$ defined using indexing I is said to be an *automatic family* if I is regular and $\{(e, x) : x \in L_e\}$ is automatic.

2.2 Transducers

A *transducer* processes its inputs at different speeds; its transition function δ does not necessarily process one symbol from each input component, but possibly none or many. Formally, a transducer is a tuple $(Q, n, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, n is input arity, Σ is a finite set of symbols, $q_0 \in Q$ is the starting state, $F \subseteq Q$ is the set of final states, and the transition function δ is a subset of $Q \times (\Sigma^*)^n \times Q$. A run of a transducer is of the form: $(p_0, s_{1,1}, \ldots, s_{1,n}, p_1), (p_1, s_{2,1}, \ldots, s_{2,n}, p_2), \ldots, (p_{k-1}, s_{k,1}, \ldots, s_{k,n}, p_k)$, where p_0 is the starting state and for all $i < k$, $(p_i, s_{i+1,1}, s_{i+1,2}, \ldots, s_{i+1,n}, p_{i+1}) \in \delta$. Note that the lengths of the $s_{i,j}$ may differ. The run is *accepting* if $p_k \in F$, and the accepted input is $(w_1, w_2, \ldots, w_n)$, where $w_i = s_{1,i} s_{2,i} \ldots s_{k,i}$ for $i = 1, 2, \ldots, n$. The relation recognised by a transducer is the set of inputs it accepts (in some accepting run); such relations are called *rational* or *transduced*. Note that the main difference between an automatic relation and a transduced relation is that

the transducer can read the inputs independently at different speeds and therefore the non-determinism of the transducer may prove useful. Nivat [20] provided a characterisation when a relation is transduced which is based on the notion of a homomorphism. Here a homomorphism is a mapping h which replaces each symbol a by a possibly empty word $h(a)$. Now a k-ary relation R is transduced iff there is a regular set A and there are k homomorphisms $h_1, \ldots, h_k$ such that, for each symbol, at most one of the homomorphism $h_1, \ldots, h_k$ maps this symbol to a non-empty word and the relation R is the set $\{(h_1(u), \ldots, h_k(u)) : u \in A\}$.

A function f is said to be *transduced* or *rational* if the relation $\{(u_1, \ldots, u_m, v_1, \ldots, v_n) : f(u_1, \ldots, u_m) = (v_1, \ldots, v_n)\}$ is rational and every input tuple $(u_1, \ldots, u_m)$ has a unique output tuple $(v_1, \ldots, v_n)$. A class of languages $\{L_e : e \in I\}$ indexed by I is said to be a *transduced family* if I is regular and the relation $\{(e, x) : x \in L_e\}$ is recognised by some transducer; the class is automatic iff the relation is recognised by a finite automaton. Transduced families have some of the decidability properties of automatic families, in particular those below.

Proposition 1. *If $\{L_e : e \in I\}$ is a transduced family and a transducer* $\mathbf{M}$ *accepts $\{(e, x) : x \in L_e, e \in I\}$, then one can effectively (from* $\mathbf{M}$*, parameter y and finite set D), for each of the following sets, find a DFA recognising it:*

(a) $A_y = \{p : y \in L_p\}$;
(b) $A'_y = \{p : y \notin L_p\}$;
(c) $B_y = \{p : \{z \in L_p : |z| > |y|\} \neq \emptyset\}$;
(d) $B'_y = \{p : \{z \in L_p : |z| > |y|\} = \emptyset\}$;
(e) $C_D = \{p : D \subseteq L_p\}$;
(f) $C'_D = \{p : D = L_p\}$;
(g) $F_e = \{x : x \in L_e\}$.

In particular, given d and e, it can be effectively determined whether $L_d \subseteq L_e$ and whether $L_d \subset L_e$.

2.3 Learning Theory

Gold [11] defined a *text* as a mapping T from $\mathbb{N}$ to $\Sigma^* \cup \{\#\}$, whose contents, denoted $content(T)$, is the set $\{T(x) : x \in \mathbb{N}\} \setminus \{\#\}$; it is a text for a language L iff $content(T) = L$. The initial segment of text T of length n is denoted $T[n]$.

To learn a target class $\mathcal{L} = \{L_e : e \in I\}$, defined using indexing I, a learner uses a hypothesis space $\mathcal{H} = \{H_e : e \in J\}$, defined using indexing J, with $\mathcal{L} \subseteq \mathcal{H}$.

The following notions are adapted from Gold [11]. A learner uses some alphabet Γ for its memory. It starts with an initial memory and hypothesis. On each datum, it updates its memory and hypothesis. That is, a *learner* is a mapping $\mathbf{M}$ from $(\Gamma^* \cup \{?\}) \times (\Sigma^* \cup \{\#\})$ to $(\Gamma^* \cup \{?\}) \times (J \cup \{?\})$, together with an initial memory mem_0 and hypothesis hyp_0. Intuitively, ? denotes both null memory (different from ε) and null hypothesis (when the learner issues no hypothesis). One can extend the definition of a learner to arbitrary initial sequences of texts T, setting $\mathbf{M}(T[0]) = (mem_0, hyp_0)$, and then inductively

setting $\mathbf{M}(T[n+1]) = (mem_{n+1}, hyp_{n+1}) = \mathbf{M}(mem_n, T(n))$. Intuitively, mem_n and hyp_n are the memory and conjecture of the learner after having seen the data in $T[n]$, respectively. A learner $\mathbf{M}$ converges on text T to a hypothesis e iff for all but finitely many n, $hyp_n = e$. Note that the memory is not required to converge. A learner $\mathbf{M}$ *explanatorily learns* a language L if for all texts T for L, $\mathbf{M}$ converges on T to a hypothesis e with $H_e = L$. A learner $\mathbf{M}$ (*explanatorily*) learns a class $\mathcal{L}$ of languages iff it learns each $L \in \mathcal{L}$ [8,9,11].

Blum and Blum [3] defined a finite sequence σ to be a *locking sequence* for a learner $\mathbf{M}$ on language L if (a) $content(\sigma) \subseteq L$, (b) the hypothesis e of $\mathbf{M}$ on σ satisfies $H_e = L$, and (c) for all τ with $\sigma \subseteq \tau$ and $content(\tau) \subseteq L$, the hypothesis of $\mathbf{M}$ on τ is e. They showed that if $\mathbf{M}$ learns L then such a σ exists.

A learner $\mathbf{M}$ is said to be recursive if the corresponding function F, mapping (old memory, datum) to (new memory, hypothesis), is recursive. Jain et al. [13] defined $\mathbf{M}$ to be an *automatic* learner if F is automatic. Finally, $\mathbf{M}$ is said to be a *transduced* learner if F can be computed by a transducer.

In this work, learners are recursive, and can or not be transduced or automatic. The memory limitations of the learner discussed in this paper is based on the length of the memory of the learner in terms of the length of the longest datum seen so far. Thus, for example, a learner $\mathbf{M}$ is word size memory bounded if for some constant c, for all finite sequences σ, if $M(\sigma) = (mem, hyp)$, and $n = \max\{|x| : x \in content(\sigma)\}$, then $|mem| \leq n + c$. Similarly, the learner is $O(n^2)$ memory bounded if $|mem| \leq cn^2 + c$.

Example 2. *For all $e \in \{0,1\}^+$, let $L_e = \{0,1\}^* \setminus (\{0,1\}^* \cdot \{e\})$ and consider the class defined by the transduced family $\{L_e : e \in \{0,1\}^+\}$. The transduced learner for this family has its current memory always the same as the current hypothesis e (initialised to 0). For an input word x, if x ends with e then e is updated to its length-lexicographic successor else e remains unchanged. Thanks to its non-deterministic nature, a transducer can check whether x ends with e and give the corresponding output. Note that both outputs, new memory and new hypothesis, of the update function of this learner are always the same.*

During the learning process, as long as the current value of e is length-lexicographically strictly below the target, the learner will eventually see an input ending with e, as there are infinitely many of these inputs, and then update the hypothesis and memory to the next binary word in length-lexicographical order. Eventually e reaches the correct value and then no further datum can cause another update of the hypothesis. Hence, one can verify that the transduced learner indeed converges to the correct hypothesis. Proposition 3 provides a more complicated version of this class that has a transduced but not an automatic learner.

3 Automatic versus Transduced Learners and Memory-Size Hierarchies for Transduced Learners

Proposition 5 shows that some automatic classes can be learnt by transduced learners but not by automatic ones. Proposition 3 shows that furthermore, if

one considers transduced classes, then a transduced learner can succeed while keeping the word-size memory bound, whereas no automatic learner succeeds.

Proposition 3. *There is a transduced family which can be learnt by a transducer with word-size memory while it does not have an automatic learner at all.*

The following result provides lower bounds for the long term memory in terms of the longest example seen so far.

Proposition 4. *Suppose that S is a regular set and $f(n) = \big|\{x \in S : |x| \leq n\}\big|$. Let $\mathcal{L} = \Big\{L : \exists n \big[L \subseteq \{x \in S : |x| \leq n\} \text{ and } f(n) - 1 \leq |L| \leq f(n)\big]\Big\}$. Then any learner, whether automatic, transduced or recursive, needs in the worst case memory of length at least $f(n)/c$, for some constant c, after having seen some sequence containing only words of length up to n.*

Proof. Consider any learner $\mathbf{M}$ for $\mathcal{L}$ which uses alphabet Γ for its memory.

For any n, consider $L = \{x \in S : |x| \leq n\}$. Suppose there are two finite sequences σ and τ with $\mathit{content}(\sigma) \neq \mathit{content}(\tau)$, $\mathit{content}(\sigma) \cup \mathit{content}(\tau) \subseteq L$, and the memories of $\mathbf{M}$ after having seen the inputs σ and τ are the same. Let z be in the symmetric difference of $\mathit{content}(\sigma)$ and $\mathit{content}(\tau)$, say in $\mathit{content}(\sigma) \setminus \mathit{content}(\tau)$. Let T be a text for $L \setminus \{z\}$. Now, $\mathbf{M}$ either converges to the same hypothesis on σT and τT or fails to converge on both. As σT and τT are texts for L and $L \setminus \{z\}$, respectively, which are different languages in $\mathcal{L}$, $\mathbf{M}$ fails to learn at least one of these languages.

It follows that $\mathbf{M}$ has at least $2^{f(n)}$ different memory values on different finite sequences with elements from S of length at most n. Thus, at least one of these memory values must have length at least $f(n)/c$, where $c = \log_2(|\Gamma|)$. □

If $f(n) = n^k$ for some constant k, the lower bound is $\Omega(n^k)$; if $f(n) = c^n$ for some constant c, the lower bound is $\Omega(c^n)$ on the maximum length of the memory on input sequences containing words of length up to n. It will be shown that for some regular languages S, similar upper bounds are obtained.

Proposition 5. *Let $S = \{0\}^* \cdot \{1\}^*$. Let $f(n) = \big|\{x \in S : |x| \leq n\}\big|$ (which is equal to $(n^2 + 3n + 2)/2$). Define $\mathcal{L}$ as the set of all languages L for which $\exists n \big[L \subseteq \{x \in S : |x| \leq n\} \text{ and } f(n) - 1 \leq |L| \leq f(n)\big]$. Then $\mathcal{L}$ can be learnt by a transduced learner with memory size $O(n^2)$ but $\mathcal{L}$ cannot be learnt by any automatic learner, even without explicit memory bounds.*

Proof. Suppose for a contradiction that an automatic learner $\mathbf{M}$ learns $\mathcal{L}$. Fixing $n \in \mathbb{N}$, consider a sequence σ containing exactly n elements of length at most n from S. As $\mathbf{M}$ is automatic, its memory on σ can be of length at most cn for some constant c, and thus the number of possible memories of $\mathbf{M}$ after seeing σ is bounded by d^n for some constant d. On the other hand, there are at least $\binom{n^2/2}{n}$ possible contents of such sequences, and for large enough n, this is larger than d^n. Thus, for large enough n, there exist two sequences σ and σ', with different content, each containing exactly n elements of length at most n, such that the

memory of $\mathbf{M}$ after seeing σ and σ' is the same. Let $x \in content(\sigma) \setminus content(\sigma')$, and let T be a text for $S \cap \{y \in S : |y| \leq n\} \setminus \{x\}$. Now, $\mathbf{M}$ on texts σT and $\sigma' T$ either does not converge or converges to the same conjecture even though they are texts for different languages in $\mathcal{L}$. Thus, $\mathbf{M}$ cannot learn $\mathcal{L}$.

Now it is shown that a transduced learner can learn $\mathcal{L}$. For representation of languages in $\mathcal{L}$, the indices are of the form $0^i1^j2^k$ and 3^{k+1}. If $w = 0^i1^j2^k$ then L_w contains all words in S of length up to $i+j+k$ except for 0^i1^j. If $w = 3^{k+1}$ then L_w contains all members of S of length up to k.

The learner uses as memory a string of the form $\{0,1\}^*$, where certain positions are marked. Intuitively, for memory $w = w(0)w(1)\ldots w(n-1)$, each position in the memory string represents a string of the form 0^i1^j, the marked positions representing the strings that have been seen in the input. A position p in the string represents the string formed by taking the number of 0s in $w(0)w(1)\ldots w(p)$, followed by $p-r$ 1s for the largest $r \leq p$ such that $w(r) = 0$; if there is no 0 in $w(0)w(1)\ldots w(p)$ then r is taken to be -1. Thus, a position p represents the string formed by taking the sequence of 0s up to position p (inclusive) followed by the number of 1s between position p (inclusive) and the position of the last 0 up to position p (inclusive).

For example, if the strings 00, 001 and 011 have been observed, then the value of the memory data structure is $011'0'1'$ and the strings represented by the positions of the marked (primed) letters as ordered in the memory word are 011, 00 and 001. Note that the principle of taking the "maximal numbers of 0s" before the third mark implies that the word 0111 is not represented in the above memory as there is a 0 between the 1s taken over. The overall goal of updating the memory is to let the current input word 0^i1^j be represented in the memory by a position and the symbol at this position be marked. The beginning of the memory can be marked in order to record that ε has been observed in the input.

The learner starts with memory ε without any marks. Now suppose that at any time, (1) the new input word is 0^i1^j, (2) i' is the number of 0s in the current memory word, and (3) j' is the number of 1s in the current memory word which have exactly i 0s in the memory before their position. Note that j' is 0 if either $i' < i$ or $i' \geq i$ and after the first i 0s, either the word ends or another 0 follows. The non-deterministic transducer does the following:

1. First, while there is a 0 to be read in both memory and input datum d: read old memory copying each symbol to new memory and whenever a 0 is read, read it also on d until at least one of the memory or d has only symbols from $\{1\}^*$ left; thus the memory is copied until $\min\{i, i'\}$ 0s (along with intermediate 1s in the memory) are copied from the old memory and d has the first $\min\{i, i'\}$ symbols read.
2. If $i' \geq i$ (this is determined by the transducer by guessing, and verifying when reading the rest of the words) then read j 1s from the current datum and j' 1s from the old memory and write $\max\{j, j'\}$ 1s to the new memory. Then copy the remaining part of the old memory to the new memory.

3. If $i' < i$ then first copy all remaining 1s from the old memory to the new memory and then copy the remaining symbols $0^{i-i'}1^j$ from the current datum to the new memory.
4. Besides this, all symbols copied from the old memory to the new memory keep their marks in case they have some already, and the symbol at the position representing 0^i1^j in the new memory also receives a mark.

What follows demonstrates how the hypothesis is written when writing to the new memory, as both outputs are independently written into different words. Still, the workings of the transducer is best understood when the transducer is thought of as non-deterministically extracting hypothesis from new memory.

Note that the memory keeps track of all words seen in the input using the marks. It can also be used to indicate missing words: if a position representing 0^i1^j is unmarked, then 0^i1^j has not been seen in the input; if the i-th 0 in the memory does not have a 1 preceding it, then $0^{i-1}1$ is not seen in the input. Call a word v stored in the memory maximal iff $v1$ is not represented in the memory. Note that if all positions in the memory are marked and the lengths of all maximal words in the memory except the maximal word v have the same parity as the length of $v1$, then $v1$ has not been seen in the input so far, even though $v1$'s length is at most that of the maximal word in the input language.

When writing the new hypothesis, the learner can verify and act according to the first of the following cases which applies. Suppose i' is the total number of 0s in the memory.

1. If the position representing ε is not marked, then the hypothesis is $2^{i'}$;
2. If the memory ends in 1, then the hypothesis is $0^{i'+1}$;
3. If some 0 in the memory is not preceded by a 1, then the hypothesis is $0^{i-1}1^{i'-i+1}$ for the least i such that the i-th 0 is not preceded by a 1;
4. If some 0 in the memory is not marked then the hypothesis is $0^i2^{i'-i}$ for the least i such that the position representing 0^i is not marked;
5. If some 1 is not marked then the hypothesis is $0^i1^j2^{j'-j}$ where j is such that 0^i1^j is represented by the position of the leftmost unmarked 1 and j' is the number of 1s between the i-th and $(i+1)$-st 0s in memory;
6. If all positions are marked and the lengths of all maximal words represented, except for the maximal word v, share the same parity, then the hypothesis is $v1$;
7. If none of the above conditions applies then the hypothesis is $3^{i'+1}$, which is the set of all words in S of length up to i'.

Note that in order to check the sixth case, the transducer can always count the number of 0s up to the current position modulo 2, and then count the number of 1s following this 0 modulo 2. Also, the transduced learner can easily verify for any particular case that none of the earlier cases applies. Thus, the learner can generate the hypothesis based on the above.

The memory keeps track of all data seen in the above described data structure. Also, the hypothesis is computed in such a way that it is correct whenever all of the shortest $f(n)$ members of S except perhaps one have been seen, but no

longer data have been observed. As an example, the following table illustrates data, memory and hypothesis updates of the learner. The initial memory is ε and the old memory is always the new memory of the previous step.

Datum	New memory	w	Words in L_w
1	$1'$	ε	None
01	$1'01'$	00	$\varepsilon, 0, 1, 01, 11$
ε	$'1'01'$	00	$\varepsilon, 0, 1, 01, 11$
00	$'1'01'0'$	02	$\varepsilon, 1, 00, 01, 11$
0	$'1'0'1'0'$	11	$\varepsilon, 0, 1, 00, 01$
11	$'1'1'0'1'0'$	333	$\varepsilon, 0, 1, 00, 01, 11$

The above example illustrates parts of the proof. □

The previous proof can be generalised to larger alphabet sizes; however, one has to fix the alphabet size and exponent of the polynomial. Thus both preceding results give the following corollary, in which the bound $\Theta(n^k)$ for the family given by $\mathcal{L}_k$ indicates that one can learn with memory size $O(n^k)$, but every learner needs at least memory size $\Omega(n^k)$.

Corollary 6. *Suppose Σ has k symbols $0, 1, \ldots, k-1$, and let S_k be $\{0\}^* \cdot \{1\}^* \cdot \ldots \cdot \{k-1\}^*$. Let $\mathcal{L}_k$ be the class of all $L_{vw} = \{x \in S_k : |x| \leq |vw| \text{ and } x \neq v\}$ for all $w \in \{k\}^*$ and $v \in \{0\}^* \cdot \{1\}^* \cdot \ldots \cdot \{k-1\}^*$, and L_w be $\{x \in S_k : |x| < |w|\}$ for all $w \in \{k+1\}^+$. Note that the number of elements in S_k of length at most n is $f(n) = \binom{n+k}{k} = \sum_{m \leq n} \binom{m+k-1}{k-1}$. Now, $\mathcal{L}_k$ can be learnt by a transduced learner with a memory length bound of $\Theta(n^k)$, where n is the length of the longest example seen so far.*

An additional corollary to Proposition 5 can be obtained with respect to target-sized learners. The result uses the following fact which Jain et al. [14] showed for all automatic families: there is a constant c such that for each language L_e, if words in the language L_e have at most length n then the shortest index d of L_e has at most length $n + c$. Stephan [25] defined a learner to have a *target-sized* memory bound if the length of the memory is never longer than the length of the shortest index of the language being learnt plus a constant. If one relaxes this bound by just requiring the existence of a function f such that the memory is never longer than $f(n)$ with n being the size of the shortest index of the target, then one can get the following corollary.

Corollary 7. *The class $\mathcal{L}_k$ (from Corollary 6) of all subsets of $\{0\}^* \cdot \{1\}^* \cdot \ldots \cdot \{k-1\}^*$ of words up to length n except perhaps one, can be learnt by a transduced learner with target-sized memory of size $O(f(n))$, where $f(n) = \binom{n+k}{k}$, but not with any better memory constraint, except for a multiplicative constant.*

The class of all languages of binary words up to length n except perhaps for one can be learnt by a transduced learner with exponential target-sized memory.

The class of all $L_w = \{v \in \Sigma^* : \varepsilon <_{ll} v \leq_{ll} w\}$ with $w \neq \varepsilon$ and $L_\varepsilon = \Sigma^*$ cannot be learnt with any type of target-sized memory.

Proposition 8. *If an automatic class can be explanatorily learnt from text then there is a transduced learner for the same class which learns it with $O(c^n)$ sized memory where c is some constant that depends only on the class and where n is the size of the longest datum seen so far.*

4 A Space Bound for Learning All Learnable Transduced Classes

A learner **M** is set-driven [21] if for all sequences σ and τ such that $content(\sigma)$ is equal to $content(\tau)$, **M**'s memory and hypothesis are the same after seeing either σ or τ. It is first shown that every learnable transduced family can be learnt by a set-driven recursive learner, which can be obtained uniformly from a transducer learner for the family. This learner is defined for all transduced families. However, for unlearnable families, the learner will fail on some input texts. The learner employs the properties of transduced families listed in Proposition 1.

A *tell-tale* set for a language L with respect to a class $\mathcal{L}$ of languages is a finite subset D of L such that for all $L' \in \mathcal{L}$, if $D \subseteq L' \subseteq L$ then $L' = L$. Angulin [1] has shown that for any learnable family of languages $\mathcal{L}$, every language in $\mathcal{L}$ has a tell-tale with respect to $\mathcal{L}$.

Proposition 9. *For every learnable transduced family $\mathcal{L} = \{L_e : e \in I\}$, some set-driven recursive learner learns that family. Furthermore, this learner can be effectively obtained from the transducer describing the transduced family.*

Hence for some recursive function g, the memory of the recursive learner is bounded by $g(n)$ where n is the length of the longest datum seen so far.

Proof. The learner is given by the following algorithm. Let $D = \{a_1, a_2, \ldots, a_m\}$ be the set of words observed in the input so far (where each a_i is distinct). Let $f(D)$ be the length-lexicographically least index e with $D \subseteq L_e$ (note that by Proposition 1, this can be found effectively). Let n be the maximum of

- the length of the description of the DFA accepting I,
- the length of the description of the transducer accepting the set defined as $\{(e, x) : x \in L_e, e \in I\}$ for the transduced family $\{L_e : e \in I\}$ to be learnt,
- the alphabet size for the transduced family and
- the lengths $|a_1|$, $|a_2|$, …, $|a_m|$ of all words in D.

Now the learner **M** chooses the first of the following options which applies:

1. If there is an $e \in I$ with $L_e = D$ then **M** outputs the length-lexicographically least such e (by Proposition 1, this is decidable for transduced families);

2. If $f(D)$ is defined, because there is e with $D \subseteq L_e$, then $\mathbf{M}$ selects the length-lexicographically least index e of length at most $|f(D)|+n$ such that $D \subseteq L_e$ and there is no index d of length at most $|f(D)|+n$ with $D \subseteq L_d \subset L_e$;
3. Otherwise, D is not consistent with any hypothesis in the family and $\mathbf{M}$ outputs ? to signal that there is no valid conjecture.

By definition, $\mathbf{M}$ is set-driven. Furthermore, if the input language L_e is finite then $\mathbf{M}$ will converge to its length-lexicographically least index after having seen all elements. If L_e is infinite then there must exist a finite subset D of L (a tell-tale set) such that there is no language L_d in $\mathcal{L}$ such that $D \subseteq L_d \subset L_e$. Thus, for large enough n, and thus after having received large enough datum, step 2 would output the least index for L_e. Thus, $\mathbf{M}$ learns all languages in $\mathcal{L}$.

For the function g, note that there are only finitely many pairs of descriptions of transduced families $\{L_e : e \in I\}$ and data sets $D \subseteq \{0,1,\ldots,n-1\}^*$ such that their size is bounded by n. One can therefore take $g(n)$ to be the maximum of the space used by the algorithm when run with the given parameterisation describing the transduced family and the data set D as input. The number $g(n)$ can be algorithmically computed from n. □

Proposition 10. *If a class has a set-driven recursive learner using space bound $g(n)$, then it also has a transduced learner using space bound $g(n)+c^n$ for its memory, for some constant c, with n denoting the size of the longest datum seen so far.*

5 Conclusion and Subsequent Work

It was demonstrated that whereas many questions on memory usage remain open for automatic learners, transduced learners that learn a transduced family can always bound the memory size as a function of the longest datum seen so far. When learning automatic families, concrete bounds have been found, and a hierarchy of polynomial and exponential bounds has emerged. It has been shown that every family can be learnt using some exponential bound.

Subsequent work addressed the question of the extent to which transduced learners can satisfy additional properties like consistency [2], conservativeness [1,22] and iterativeness [26]. Many learners constructed here can be made iterative; furthermore, transduced learners can be made consistent and conservative, similarly to the polynomial-time setting [7]; however, these criteria cannot be combined with more restrictive memory-limitations. These results have been delayed to the full version of the paper due to space limitations.

References

1. Angluin, D.: Inductive inference of formal languages from positive data. Inf. Control **45**, 117–135 (1980)
2. Bārzdiņš, J.: Inductive inference of automata, functions and programs. In: Proceedings of the 20th International Congress of Mathematicians, Vancouver, pp. 455–460 (1974). (in Russian). English translation in American Mathematical Society Translations: Series 2, 109:107–112 (1977)
3. Blum, L., Blum, M.: Toward a mathematical theory of inductive inference. Inf. Control **28**, 125–155 (1975)
4. Blumensath, A., Grädel, E.: Automatic structures. In: 15th Annual IEEE Symposium on Logic in Computer Science, LICS 2000. pp. 51–62 (2000)
5. Case, J., Jain, S., Le, T.D., Ong, Y.S., Semukhin, P., Stephan, F.: Automatic learning of subclasses of pattern languages. In: Dediu, A.-H., Inenaga, S., Martín-Vide, C. (eds.) LATA 2011. LNCS, vol. 6638, pp. 192–203. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21254-3_14
6. Case, J., Jain, S., Seah, S., Stephan, F.: Automatic functions, linear time and learning. Log. Methods Comput. Sci. **9**(3:19), 1–26 (2013)
7. Case, J., Kötzing, T.: Difficulties in forcing fairness of polynomial time inductive inference. In: Gavaldà, R., Lugosi, G., Zeugmann, T., Zilles, S. (eds.) ALT 2009. LNCS (LNAI), vol. 5809, pp. 263–277. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04414-4_23
8. Case, J., Lynes, C.: Machine inductive inference and language identification. In: Nielsen, M., Schmidt, E.M. (eds.) ICALP 1982. LNCS, vol. 140, pp. 107–115. Springer, Heidelberg (1982). https://doi.org/10.1007/BFb0012761
9. Case, J., Smith, C.: Comparison of identification criteria for machine inductive inference. Theor. Comput. Sci. **25**, 193–220 (1983)
10. Freivalds, R., Kinber, E., Smith, C.H.: On the impact of forgetting on learning machines. J. ACM **42**, 1146–1168 (1995)
11. Gold, E.M.: Language identification in the limit. Inf. Control **10**, 447–474 (1967)
12. Hodgson, B.R.: Théories décidables par automate fini. Ph.D. thesis, Département de mathématiques et de statistique, Université de Montréal (1976)
13. Jain, S., Luo, Q., Stephan, F.: Learnability of automatic classes. J. Comput. Syst. Sci. **78**, 1910–1927 (2012)
14. Jain, S., Ong, Y.S., Pu, S., Stephan, F.: On automatic families. In: Proceedings of the Eleventh Asian Logic Conference, in Honor of Professor Chong Chitat on his Sixtieth Birthday, pp. 94–113. World Scientific (2012)
15. Jain, S., Osherson, D.N., Royer, J.S., Sharma, A.: Systems That Learn, 2nd edn. Bradford—MIT Press, Cambridge (1999)
16. Khoussainov, B., Nerode, A.: Automatic presentations of structures. In: Leivant, D. (ed.) LCC 1994. LNCS, vol. 960, pp. 367–392. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60178-3_93
17. Khoussainov, B., Nies, A., Rubin, S., Stephan, F.: Automatic structures: richness and limitations. Log. Methods Comput. Sci. **3**(2:2), 1–18 (2017)
18. Kinber, E., Stephan, F.: Language learning from texts: mind changes, limited memory and monotonicity. Inf. Comput. **123**, 224–241 (1995)
19. Lange, S., Zeugmann, T., Zilles, S.: Learning indexed families of recursive languages from positive data: a survey. Theor. Comput. Sci. **397**, 194–232 (2008)
20. Nivat, M.: Transductions des langages de Chomsky. Annales de l'Institut Fourier, Grenoble **18**, 339–455 (1968)

21. Osherson, D., Stob, M., Weinstein, S.: Learning strategies. Inf. Control **53**, 32–51 (1982)
22. Osherson, D., Stob, M., Weinstein, S.: Systems That Learn. An introduction to learning theory for cognitive and computer scientists. Bradford—MIT Press, Cambridge (1986)
23. Pitt, L.: Inductive inference, DFAs, and computational complexity. In: Jantke, K.P. (ed.) AII 1989. LNCS, vol. 397, pp. 18–44. Springer, Heidelberg (1989). https://doi.org/10.1007/3-540-51734-0_50
24. Rubin, S.: Automata presenting structures: a survey of the finite string case. Bull. Symb. Log. **14**, 169–209 (2008)
25. Stephan, F.: Methods and theory of automata and languages. School of Computing, National University of Singapore (2016)
26. Wiehagen, R.: Limes-erkennung rekursiver funktionen durch spezielle strategien. J. Inf. Process. Cybern. (EIK) **12**(1–2), 93–99 (1976)