Advanced Automata Theory 11 Regular Languages and Learning Theory

Frank Stephan

Department of Computer Science

Department of Mathematics

National University of Singapore

fstephan@comp.nus.edu.sg

Repetition: Rational Relations

Automatic Relation: Finite automaton reads all inputs involved at the same speed with # supplied for exhausted inputs.

Rational Relation: Nondeterministic finite automaton reads all inputs individually and can read them at different speed.

The first type of automata is called synchronous, the second type is called asynchronous.

There are many relations which are rational but not automatic.

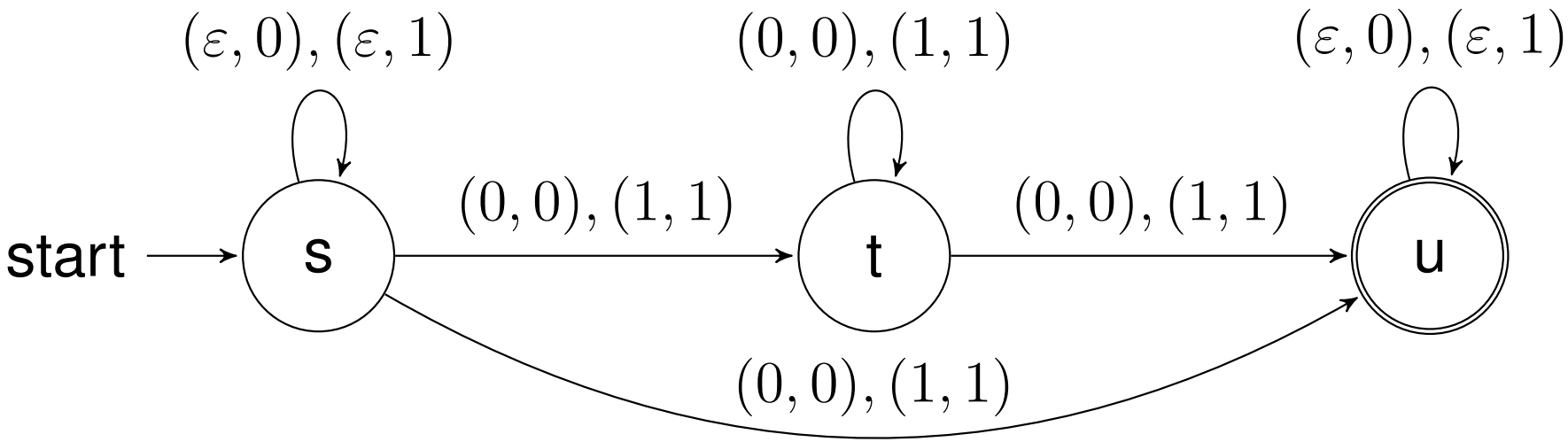
Repetition: Formal definition

A rational relation $\mathbf{R} \subseteq (\Sigma^*)^{\mathbf{n}}$ is given by an non-deterministic finite state machine which can process $\mathbf{n}$ inputs in parallel and does not need to read them in the same speed. Transitions from one state $\mathbf{p}$ to a state $\mathbf{q}$ are labelled with an $\mathbf{n}$-tuple $(\mathbf{w_1}, \mathbf{w_2}, \ldots, \mathbf{w_n})$ of words $\mathbf{w_1}, \mathbf{w_2}, \ldots, \mathbf{w_n} \in \Sigma^*$ and the automaton can go along this transition iff for each input $\mathbf{k}$ the next $|\mathbf{w_k}|$ symbols in the input are exactly those in the string $\mathbf{w_k}$ (this condition is void if $\mathbf{w_k} = \varepsilon$) and in the case that the automaton goes on this transition, $|\mathbf{w_k}|$ symbols are read from the $\mathbf{k}$-th input word.

A word $(\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n})$ is in $\mathbf{R}$ iff there is a run of the machine with transitions labelled by $(\mathbf{w_{1,1}}, \mathbf{w_{1,2}}, \ldots, \mathbf{w_{1,n}})$, $(\mathbf{w_{2,1}}, \mathbf{w_{2,2}}, \ldots, \mathbf{w_{2,n}})$, …, $(\mathbf{w_{m,1}}, \mathbf{w_{m,2}}, \ldots, \mathbf{w_{m,n}})$ ending up in an accepting state such that $\mathbf{x_1} = \mathbf{w_{1,1}w_{2,1}} \ldots \mathbf{w_{m,1}}$, $\mathbf{x_2} = \mathbf{w_{2,1}w_{2,2}} \ldots \mathbf{w_{m,2}}$, …, $\mathbf{x_n} = \mathbf{w_{1,n}w_{2,n}} \ldots \mathbf{w_{m,n}}$.

Repetition: Example 10.4

The following automaton recognises the relation of all $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x}$ is a nonempty substring of $\mathbf{y}$, that is, $\mathbf{x} \neq \varepsilon$ and $\mathbf{y} = \mathbf{vxw}$ for some $\mathbf{v}, \mathbf{w} \in \{\mathbf{0}, \mathbf{1}\}^*$.



In $\mathbf{s}$: Parsing $(\varepsilon, \mathbf{v})$;
From $\mathbf{s}$ to $\mathbf{u}$: Parsing $(\mathbf{x}, \mathbf{x})$;
In $\mathbf{u}$: Parsing $(\varepsilon, \mathbf{w})$.

Repetition: Transducers

Transducers are automata recognising rational relations where the first parameter $\mathbf{x}$ is called input and the second parameter $\mathbf{y}$ is called output.

In particular one is interested in transducers computing functions: Two possible runs accepting $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}, \mathbf{z})$ must satisfy $\mathbf{y} = \mathbf{z}$. Given $\mathbf{x}$, there is a run accepting some pair $(\mathbf{x}, \mathbf{y})$ iff $\mathbf{x}$ is in the domain of the function.

Two main concpets:

Mealy machines:
Input/Output pairs on transition-labels.

Moore machines:
Input on transition-labels; output in states.

Learning Theory

Formal models of learning processes

Query Learning

Dialogue between learner (pupil) and teacher.
Learner asks queries using a specific query language.
Teacher provides answers to these queries.

Learning from Data

Learner reads more and more data about the concept to be learnt.
In response to the data, learner conjectures one or several hypotheses.
The last hypothesis conjectured must be correct.

Angluin's DFA Learner

Query language – Learner can ask the following questions:

- Equivalence query (EQ): Does this DFA recognise the language $\mathbf{L}$ to be learnt?
  Teacher says "YES" or "NO"; when saying "NO" an $\mathbf{x}$ is provided on which the DFA does the opposite of what $\mathbf{L}$ does.
- Membership query (MQ): Is $\mathbf{x}$ a member of $\mathbf{L}$?
  Teacher says "YES" or "NO".

Learner could use a list of all DFAs and ask: does the first DFA recognise $\mathbf{L}$; does the second DFA recognise $\mathbf{L}$; ...?

This needs more than $\mathbf{2^n}$ steps in order to deal with all languages recognised by DFA's with $\mathbf{n}$ states.

Angluin's algorithm needs much less queries.

Data of Angluin's algorithm

Represent set $\mathbf{S}$ of states by strings $\mathbf{x} \in \mathbf{S}$ on which automaton goes into corresponding states. $\mathbf{x} \in \mathbf{S}$ is an accepting state iff $\mathbf{x} \in \mathbf{L}$.

Represent set $\mathbf{E}$ of observations so large that for each distinct $\mathbf{x}, \mathbf{y} \in \mathbf{S}$ there is an $\mathbf{z} \in \mathbf{E}$ with $\mathbf{L}(\mathbf{xz}) \neq \mathbf{L}(\mathbf{yz})$; that is $\mathbf{row}^{\mathbf{E}}(\mathbf{x}) \neq \mathbf{row}^{\mathbf{E}}(\mathbf{y})$ where $\mathbf{row}^{\mathbf{E}}(\mathbf{u}) = \{(\mathbf{z}, \mathbf{T}(\mathbf{uz})) : \mathbf{z} \in \mathbf{E}\}$.

Make a transition from $\mathbf{x} \in \mathbf{S}$ on symbol $\mathbf{a} \in \mathbf{\Sigma}$ to the $\mathbf{y} \in \mathbf{S}$ with $\mathbf{row}^{\mathbf{E}}(\mathbf{y}) = \mathbf{row}^{\mathbf{E}}(\mathbf{xa})$.

Maintain for this a table $\mathbf{T}$ of observations such that for all $\mathbf{w} \in \mathbf{S} \cdot \mathbf{E} \cup \mathbf{S} \cdot \mathbf{\Sigma} \cdot \mathbf{E}$, the entry $\mathbf{T}(\mathbf{w})$ is equal to $\mathbf{L}(\mathbf{w})$.

$\mathbf{DFA}(\mathbf{S}, \mathbf{E}, \mathbf{T})$ describes the DFA given by $\mathbf{S}, \mathbf{\Sigma}, \mathbf{E}, \mathbf{T}$ as above.

The Algorithm

Angluin's Algorithm (Slightly Modified) to learn unknown regular language $\mathbf{L}$ using MQs and EQs.

1. Let $\mathbf{S} = \{\varepsilon\}$ and $\mathbf{E} = \{\varepsilon\}$;
2. Make MQs to determine $\mathbf{L(w)}$ for all $\mathbf{w} \in \mathbf{S} \cdot (\mathbf{\Sigma} \cup \{\varepsilon\}) \cdot \mathbf{E}$ and let $\mathbf{T(w)} = \mathbf{L(w)}$ (where not done before);
3. Let $\mathbf{row^E(u)} = \{(\mathbf{z}, \mathbf{T(uz)}) : \mathbf{z} \in \mathbf{E}\}$ for all $\mathbf{u} \in \mathbf{S} \cdot (\mathbf{\Sigma} \cup \{\varepsilon\})$ and search for an $\mathbf{u} \in \mathbf{S}$ and $\mathbf{a} \in \mathbf{\Sigma}$ with $\mathbf{row^E(ua)} \neq \mathbf{row^E(u')}$ for all $\mathbf{u'} \in \mathbf{S}$;
   if found then let $\mathbf{S} = \mathbf{S} \cup \{\mathbf{ua}\}$ and go to 2 else go to 4;
4. Make EQ whether $\mathbf{DFA(S, E, T)}$ is correct;
5. If the answer is "YES" then terminate;
6. If the answer is "NO" with counter example $\mathbf{w}$ then let $\mathbf{E} = \mathbf{E} \cup \{\mathbf{v} : \exists \mathbf{u} [\mathbf{w} = \mathbf{uv}]\}$ and go to 2.

Verification

Let $(\mathbf{Q}, \mathbf{\Sigma}, \delta, \mathbf{s}, \mathbf{F})$ be the smallest dfa for $\mathbf{L}$.

(A) If $\mathbf{x}, \mathbf{y} \in \mathbf{S}$ are distinct then $\delta(\mathbf{s}, \mathbf{x}) \neq \delta(\mathbf{s}, \mathbf{y})$.

(B) If an equivalence query is made then a new state is added eventually.

(C) Angluin's learner uses at most $|\mathbf{Q}|$ equivalence queries.

(D) When reaching line 4, $\mathbf{DFA}(\mathbf{S}, \mathbf{E}, \mathbf{T})$ is complete, that is, for every $\mathbf{u} \in \mathbf{S}$ and $\mathbf{a} \in \mathbf{\Sigma}$ there is a $\mathbf{u'} \in \mathbf{S}$ with $\mathbf{row^E(u')} = \mathbf{row^E(ua)}$.

(E) If $\mathbf{z} \in \mathbf{E}$ then $\mathbf{DFA}(\mathbf{S}, \mathbf{E}, \mathbf{T})$ accepts $\mathbf{z}$ iff $\mathbf{T}(\mathbf{z}) = \mathbf{1}$.

(F) The number of membership queries is bounded by $|\mathbf{Q}| \cdot (\mathbf{r} + \mathbf{1}) \cdot (|\mathbf{\Sigma}| + \mathbf{1})$ where $\mathbf{r}$ is the sum of the lengthes of all counter examples.

Original Algorithm

Teacher has regular set $\mathbf{L}$ and learner makes membership and equivalence queries.

1. Initialise $\mathbf{S} = \{\varepsilon\}$ and $\mathbf{E} = \{\varepsilon\}$.
2. For all $\mathbf{w} \in \mathbf{S} \cdot \mathbf{E} \cup \mathbf{S} \cdot \mathbf{\Sigma} \cdot \mathbf{E}$ where $\mathbf{T}(\mathbf{w})$ is undefined, make MQ to find $\mathbf{L}(\mathbf{w})$ and let $\mathbf{T}(\mathbf{w}) = \mathbf{L}(\mathbf{w})$.
3. If there are $\mathbf{u}, \mathbf{u}' \in \mathbf{S}$, $\mathbf{a} \in \mathbf{\Sigma}$ and $\mathbf{v} \in \mathbf{E}$ such that $\mathbf{row^E(u)} = \mathbf{row^E(u')}$ and $\mathbf{T(uav)} \neq \mathbf{T(u'av)}$ then let $\mathbf{E} = \mathbf{E} \cup \{\mathbf{av}\}$ and go to 2.
4. If there are $\mathbf{u} \in \mathbf{S}$ and $\mathbf{a} \in \mathbf{\Sigma}$ with $\mathbf{row^E(ua)} \neq \mathbf{row^E(u')}$ for all $\mathbf{u}' \in \mathbf{S}$ then let $\mathbf{S} = \mathbf{S} \cup \{\mathbf{ua}\}$ and go to 2.
5. Make an EQ whether $\mathbf{DFA(S, E, T)}$ recognises $\mathbf{L}$.
6. If the answer is “YES” then terminate with $\mathbf{DFA(S, E, T)}$.
7. If the answer is “NO” with counterexample $\mathbf{w}$ then let $\mathbf{S} = \mathbf{S} \cup \{\mathbf{u} : \exists \mathbf{v} \, [\mathbf{uv} = \mathbf{w}]\}$ and go to 2.

Learning from Positive Data

A text $\mathbf{T}$ for $\mathbf{L}$ is an infinite sequence $\mathbf{T(0), T(1), \ldots}$ of strings and pause symbols $\#$ describing $\mathbf{L}$: $\mathbf{w \in L}$ iff $\mathbf{w \neq \#}$ and $\mathbf{w = T(n)}$ for some $\mathbf{n}$.

Learner $\mathbf{M}$ starts with initial memory $\mathbf{mem_0}$ and hypothesis $\mathbf{e_0}$. In the $\mathbf{n}$-th cycle, $\mathbf{M}$ updates $\mathbf{mem_n}$ and input $\mathbf{T(n)}$ to $\mathbf{mem_{n+1}}$ and $\mathbf{e_{n+1}}$; in short: $\mathbf{M}$ maps $\mathbf{(mem_n, T(n))}$ to $\mathbf{(mem_{n+1}, e_{n+1})}$. The values $\mathbf{e_1, e_2, \ldots}$ depend on $\mathbf{T}$.

Learner $\mathbf{M}$ learns $\mathbf{L}$ iff, on every text $\mathbf{T}$ for $\mathbf{L}$, almost all hypotheses $\mathbf{e_n}$ of $\mathbf{M}$ are the same $\mathbf{e}$ describing $\mathbf{L}$.

Example

Let $\mathbf{I} = \{\mathbf{0}\}^*$ and $\mathbf{L_e} = \{\mathbf{x} \in \{\mathbf{0},\mathbf{1}\}^* : |\mathbf{x}| < |\mathbf{e}|\}$. For example, $\mathbf{L_{00}} = \{\varepsilon, \mathbf{0}, \mathbf{1}\}$ and $\mathbf{L_{000}} = \{\varepsilon, \mathbf{0}, \mathbf{1}, \mathbf{00}, \mathbf{01}, \mathbf{10}, \mathbf{11}\}$.

The class $\{\mathbf{L_e} : \mathbf{e} \in \mathbf{I}\}$ can be learnt from positive data.

$\mathbf{mem_n}$ is the longest datum seen before round $\mathbf{n}$ and $\mathbf{mem_n} = \#$ if no datum has been seen before round $\mathbf{n}$.

If $\mathbf{mem_n} = \#$ then $\mathbf{e_n} = \varepsilon$ else $\mathbf{e_n} = \mathbf{0^{k+1}}$ for the length $\mathbf{k}$ of $\mathbf{mem_n}$.

When learning $\mathbf{L_{0^k}}$ then after some time some string of length $\mathbf{k-1}$ has shown up in the input and from then onwards, $\mathbf{mem_n}$ is that string and $\mathbf{e_n} = \mathbf{0^k}$. So the learner eventually converges to the right hypothesis.

Next slide: Hypotheses made for data from sample text $\#, \#, \varepsilon, \mathbf{11}, \mathbf{01}, \mathbf{0}, \mathbf{1}, \ldots$ and the corresponding memory.

Sample Text

n	first n members of T	mem_n	e_n	L_{e_n}
0	–	$\#$	ε	$\emptyset$
1	$\#$	$\#$	ε	$\emptyset$
2	$\#, \#$	$\#$	ε	$\emptyset$
3	$\#, \#, \varepsilon$	ε	0	$\{\varepsilon\}$
4	$\#, \#, \varepsilon, 11$	11	000	$\{\varepsilon, 0, 1, 00, 01, 10, 11\}$
5	$\#, \#, \varepsilon, 11, 01$	11	000	$\{\varepsilon, 0, 1, 00, 01, 10, 11\}$
6	$\#, \#, \varepsilon, 11, 01, 0$	11	000	$\{\varepsilon, 0, 1, 00, 01, 10, 11\}$
7	$\#, \#, \varepsilon, 11, 01, 0, 1$	11	000	$\{\varepsilon, 0, 1, 00, 01, 10, 11\}$
8	$\#, \#, \varepsilon, 11, 01, 0, 1, 101$	101	0000	$\{\varepsilon, 0, 1, \ldots, 111\}$

If all data is from $\mathbf{L_{0000}}$ then the learner keeps the hypothesis $\mathbf{0000}$ forever.

Learnability of Classes

Theorem [Gold 1967]
The class of all regular languages cannot be learnt from positive data.

Theorem [Angluin 1980]
An automatic family $\{\mathbf{L_e} : \mathbf{e} \in \mathbf{I}\}$ can be learnt from positive data iff there is for every $\mathbf{e} \in \mathbf{I}$ a finite subset $\mathbf{F_e} \subseteq \mathbf{L_e}$ such that there is no $\mathbf{d} \in \mathbf{I}$ with $\mathbf{F_e} \subseteq \mathbf{L_d} \subset \mathbf{L_e}$.

Example
The class of $\mathbf{L}_\varepsilon = \mathbf{\Sigma}^*$ and $\mathbf{L_x} = \{\mathbf{y} \in \mathbf{\Sigma}^* : |\mathbf{y}| < |\mathbf{x}|\}$ with $\mathbf{x} \in \mathbf{\Sigma}^+$ does not satisfy Angluin's tell-tale condition; hence this class is not learnable.

Learner for Tell-Tale Condidtion

Let $\{\mathbf{L_e} : \mathbf{e} \in \mathbf{I}\}$ be a given automatic class.

Let $\mathbf{F_e}$ be the tell-tale of $\mathbf{L_e}$.

Now $\mathbf{mem_n}$ is the set of data observed before step $\mathbf{n}$ and updated by $\mathbf{mem_{n+1}} = \mathbf{mem_n} \cup \{\mathbf{T(n)}\}$.

Furthermore, $\mathbf{e_n}$ is the length-lexicographically least $\mathbf{e} \in \mathbf{I}$ with $\mathbf{F_e} \subseteq \mathbf{mem_n} \subseteq \mathbf{L_e}$; if such an $\mathbf{e}$ does not exist then $\mathbf{e_n}$ is a default index of $\emptyset$.

When learning $\mathbf{L_e}$, after sufficient time the learner has seen all data of $\mathbf{F_e}$ and for each $\mathbf{d} <_{\mathbf{ll}} \mathbf{e}$, either $\mathbf{F_d} \not\subseteq \mathbf{L_e}$ or some datum in $\mathbf{L_e} - \mathbf{L_d}$ has been observed.

Hence the learner will converge to $\mathbf{e}$.

Necessity of Tell-Tale Condition

Let $\{\mathbf{L_e} : \mathbf{e} \in \mathbf{I}\}$ be a given automatic class and assume that there is an $\mathbf{e} \in \mathbf{I}$ such that for $\mathbf{L_e}$ there does not exist a finite tell-tale set.

Assume that $\mathbf{M}$ is a learner for the class.

Goal: Make a text $\mathbf{T}$ for $\mathbf{L_e}$ on which $\mathbf{M}$ outputs infinitely often a wrong conjecture.

Idea is to construct the text $\mathbf{T}$ inductively, starting with $\mathbf{n = 0}$ and alternating between 1 and 2:
1. Let $\mathbf{T(n) = min_{ll}(L_e - \{T(m) : m < n\})}$, $\mathbf{n = n + 1}$;
2. Choose some $\mathbf{L_d}$ with $\mathbf{\{T(m) : m < n\} \subseteq L_d \subset L_e}$ and keep updating $\mathbf{T(n) = min_{ll}(L_d - \{T(m) : m < n\})}$, $\mathbf{n = n + 1}$ until the conjecture $\mathbf{e_n}$ of $\mathbf{M}$ equals $\mathbf{d}$.

Automatic Learners

Automatic learner given by $\mathbf{mem_0}, \mathbf{e_0}$ and
$\mathbf{uf} : \mathbf{conv}(\mathbf{mem_n}, \mathbf{x_n}) \mapsto \mathbf{conv}(\mathbf{mem_{n+1}}, \mathbf{e_{n+1}})$.
$\mathbf{mem_n}$ is a string in an arbitrary alphabet;
$\mathbf{e_n}$ is a hypothesis from $\mathbf{I}$ when learning $\{\mathbf{L_e} : \mathbf{e} \in \mathbf{I}\}$;
$\mathbf{uf}$ is an automatic update function of the learner.

Automatic learners can only remember few information about the past, as there is a constant $\mathbf{c}$ with $|\mathbf{mem_{n+1}}| \leq \max\{|\mathbf{mem_n}|, |\mathbf{x_n}|\} + \mathbf{c}$ for all $\mathbf{n}$.

Explicit memory bounds (with constant $\mathbf{k}$) when learning $\mathbf{L_d}$:

- Word-sized: $|\mathbf{mem_{n+1}}| \leq \max\{|\mathbf{x_0}|, |\mathbf{x_1}|, \ldots, |\mathbf{x_n}|\} + \mathbf{k}$.
- Hypothesis-sized: $|\mathbf{mem_{n+1}}| \leq |\mathbf{e_{n+1}}| + \mathbf{k}$.
- Target-sized: $|\mathbf{mem_{n+1}}| \leq |\mathbf{d}| + \mathbf{k}$.
- Constant: $|\mathbf{mem_{n+1}}| \leq \mathbf{k}$.

Example 11.6

Assume $\mathbf{L_0} = \emptyset$, $\mathbf{L_1} = \{0\}^*$, $\mathbf{L_2} = \{1\}^*$ and $\mathbf{L_3} = \{0, 1\}^*$.

This finite class can be learnt by tracking whether 0 and 1 have shown up in the input:

If neither 0 nor 1 are observed: $\mathbf{L_0}$;
If 0 but not 1 is observed: $\mathbf{L_1}$;
If 1 but not 0 is observed: $\mathbf{L_2}$;
If both 0 and 1 are observed: $\mathbf{L_3}$.

This principle generalises to any finite class and learner needs only constant memory.

Every finite class of sets has an automatic learner.

Example 11.7

Assume $\Sigma = \{0, 1, 2\}$ and
$\mathbf{I} = \{\mathbf{conv}(\mathbf{v}, \mathbf{w}) : \mathbf{v}, \mathbf{w} \in \Sigma^* \wedge \mathbf{v} \leq_{\mathbf{lex}} \mathbf{w}\} \cup \{\mathbf{conv}(\mathbf{3}, \mathbf{3})\}$ with
$\mathbf{L}_{\mathbf{conv(v,w)}} = \{\mathbf{u} \in \Sigma^* : \mathbf{v} \leq_{\mathbf{lex}} \mathbf{u} \leq_{\mathbf{lex}} \mathbf{w}\}$ for all
$\mathbf{conv}(\mathbf{v}, \mathbf{w}) \in \mathbf{I}$; note that $\mathbf{L}_{\mathbf{conv(3,3)}} = \emptyset$.

Data seen so far	Hypothesis	Conjectured language
—	conv(3,3)	$\emptyset$
#	conv(3,3)	$\emptyset$
# 00	conv(00,00)	$\{00\}$
# 00 0000	conv(00,0000)	$\{00, 000, 0000\}$
# 00 0000 1	conv(00,1)	$\{u : 00 \leq_{lex} u \leq_{lex} 1\}$
# 00 0000 1 0	conv(0,1)	$\{u : 0 \leq_{lex} u \leq_{lex} 1\}$
# 00 0000 1 0 112	conv(0,112)	$\{u : 0 \leq_{lex} u \leq_{lex} 112\}$
# 00 0000 1 0 112 #	conv(0,112)	$\{u : 0 \leq_{lex} u \leq_{lex} 112\}$

Exercises 11.8 and 11.9

Exercise 11.8

Make an automatic learner which learns the class of all $\mathbf{L_d} = \{\mathbf{dw} : \mathbf{w} \in \mathbf{\Sigma}^*\}$ with $\mathbf{d} \in \mathbf{\Sigma}^*$; that is, $\mathbf{I} = \mathbf{\Sigma}^*$ in this case.

Exercise 11.9

Let $\{\mathbf{L_d} : \mathbf{d} \in \mathbf{I}\}$ be given with $\mathbf{L_d} \neq \mathbf{L_{d'}}$ whenever $\mathbf{d}, \mathbf{d'} \in \mathbf{I}$ are different. Assume that an automatic learner uses this class as a hypothesis space for learning satisfying any of the above memory constraints. Let $\{\mathbf{H_e} : \mathbf{e} \in \mathbf{J}\}$ be any other automatic family containing $\{\mathbf{L_d} : \mathbf{d} \in \mathbf{I}\}$ as a subclass. Show that there is an automatic learner satisfying the same type of memory constraints conjecturing indices taken from $\mathbf{J}$ in place of $\mathbf{I}$.

Separation

Theorem 11.10

Let $\mathbf{I} = \mathbf{\Sigma}^*$, $\mathbf{L}_\varepsilon = \mathbf{\Sigma}^+$ and $\mathbf{L_d} = \{\mathbf{w} \in \mathbf{\Sigma}^* : \mathbf{w} <_{\mathbf{ll}} \mathbf{d}\}$ for $\mathbf{d} \in \mathbf{\Sigma}^+$. The class $\{\mathbf{L_d} : \mathbf{d} \in \mathbf{I}\}$ can be learnt using a word-sized memory but not using an hypothesis-sized memory.

Word-sized Learner

Memory are length-lexicographically smallest and largest words seen so far.
If ε has not been seen, conjecture $\mathbf{\Sigma}^+$. If ε and $\mathbf{w}$ are minimal and maximal word seen so far then let $\mathbf{d}$ be the successor of $\mathbf{w}$ and conjecture $\mathbf{L_d}$.

Theorems 11.11 and 11.12

Theorem 11.11

There is a class which can be learnt with hypothesis-sized memory but not with target-sized memory and not with constant memory.

Theorem 11.12

If a learner learns a class with target-sized memory then the learner's memory is also word-sized on texts for languages in the class.
In particular, if a class has an automatic learner with a target-sized memory-limitation then it also has an automatic learner with a word-sized memory-limitation.

Exercise 11.14

Assume that $\{\mathbf{L_d} : \mathbf{d} \in \mathbf{I}\}$ is the class to be learnt and that every language in the class is finite and that for every language in the class there is exactly one index in $\mathbf{I}$.

Show that if there is a learner using word-sized memory for this class, then the memory of the same learner is also target-sized. For this, show that there is a constant $\mathbf{k}$ such that all $\mathbf{d} \in \mathbf{I}$ and $\mathbf{x} \in \mathbf{L_d}$ satisfy $|\mathbf{x}| \leq |\mathbf{d}| + \mathbf{k}$ and then deduce the full result.

Iterative Learning

A learner is iterative iff $\mathbf{mem_n = e_n}$, that is, the memory is exactly the most recent hypothesis.

Exercise 11.15

Show that there is an automatic family $\{\mathbf{L_d} : \mathbf{d} \in \mathbf{I}\}$ such that $\mathbf{I}$ contains for each $\mathbf{L_d}$ exactly one index and the $\mathbf{L_d}$ are exactly the finite subsets of $\{\mathbf{0}\}^*$ with even cardinality.

Show that the class $\{\mathbf{L_d} : \mathbf{d} \in \mathbf{I}\}$ has an iterative learner using the given hypothesis space.

Is the same possible when the class consists of all subsets of $\{\mathbf{0}\}^*$ with $\mathbf{0}$ or $\mathbf{3}$ or $\mathbf{4}$ elements?

Note that an iterative learner which just conjectured an $\mathbf{d} \in \mathbf{I}$ must abstain from updating the hypothesis on any datum $\mathbf{x} \in \mathbf{L_d}$.

Automatic Families

Which of the following classes can be represented as automatic families? If they are automatic then provide the corresponding coding including the index set else write why they cannot be automatic families.

Exercise 11.16: The family of all finite subsets of $\{0\}^* \cdot \{1\}^*$.

Exercise 11.17: The family of an infinite regular set L and all subsets of up to 5 elements.

Exercise 11.18: The family of all sets of decimal numbers which, for some $n > 0$, contain exactly two digits each n times and all other digits 0 times.

Learners

Exercise 11.19: Consider an automatic family $\{\mathbf{L_e} : \mathbf{e} \in \mathbf{I}\}$ which satisfies for each two distinct $\mathbf{d}, \mathbf{e} \in \mathbf{I}$ that either $\mathbf{L_d} \subset \mathbf{L_e}$ or $\mathbf{L_e} \subset \mathbf{L_d}$ and in which there is for each $\mathbf{e}$ a unique $\mathbf{x_e}$ such that $\mathbf{x_e} \in \mathbf{L_e}$ but $\mathbf{x_e} \notin \mathbf{L_d}$ for any $\mathbf{d}$ with $\mathbf{L_d} \subset \mathbf{L_e}$. Prove that the mapping $\mathbf{e} \mapsto \mathbf{x_e}$ is automatic and construct an automatic learner for this family.

Exercise 11.20: Given $\{\mathbf{L_d} : \mathbf{d} \in \mathbf{I}\}$ and $\mathbf{L_e} : \mathbf{e} \in \mathbf{J}\}$ both satisfying the specification of Exercise 11.19, construct an automatic learner for the family of all $\mathbf{K_{conv(d,e)}}$ with $\mathbf{d} \in \mathbf{I}, \mathbf{e} \in \mathbf{J}$ and $\mathbf{K_{conv(d,e)}} = \{\mathbf{0x} : \mathbf{x} \in \mathbf{L_d}\} \cup \{\mathbf{1y} : \mathbf{y} \in \mathbf{H_e}\}$.

Open Problems

1. Does every automatic family which has an automatic learner also have a learner with word-sized memory?
2. Does every automatic family which has a learner with hypothesis-sized memory also have a learner with word-sized memory?
3. Does every automatic family which has a learner with hypothesis-sized memory also have an iterative learner?