Omega Automata: Minimization and Learning[1]

Oded Maler

CNRS - VERIMAG
Grenoble, France

2007

[1] Joint work with A. Pnueli, late 80s

Summary

- Machine learning in general and of formal languages in particular
- States, minimization and learning in finitary automata
- Basics of ω-automata
- Why minimization/learning does not work for ω-languages in the general case
- A solution for the $\mathbf{B} \cap \bar{\mathbf{B}}$ subclass
- Toward a general solution

Machine Learning

- Given a sample consisting of a set of pairs $(x, f(x))$ for some unknown function f
- Find a (representation of) a function $f' : X \to Y$ which is compatible with the sample

Machine Learning

- Given a sample consisting of a set of pairs $(x, f(x))$ for some unknown function f
- Find a (representation of) a function $f' : X \rightarrow Y$ which is compatible with the sample
- Many issues and variations:
  - Validity of inductive inference
  - Static or dynamic sampling
  - Passive or active sampling - can we influence the choice of examples
  - Evaluation criteria: identification in the limit, probabilities, etc.

Learning Formal Languages

- For sets of sequences (languages) $L \subseteq \Sigma^*$, we want to learn the characteristic function $\chi_L : \Sigma^* \to \{0, 1\}$
- The sample elements are of the form $(u, \chi_L(u))$
- The goal is to find a representation (say, automaton) compatible with the sample

Learning Formal Languages

- For sets of sequences (languages) $L \subseteq \Sigma^*$, we want to learn the characteristic function $\chi_L : \Sigma^* \to \{0, 1\}$
- The sample elements are of the form $(u, \chi_L(u))$
- The goal is to find a representation (say, automaton) compatible with the sample
- The problem was first posed in *Moore 56: Gedanken experiments on sequential machines*
- It was solved in *Gold 72: System identification via state characterization*
- Various complexity issues concerning the number of examples as a function of the number of states (Gold, Trakhtenbrot and Barzdins, Angluin)

Regular Sets and their Syntactic Congruences

- With every $L \subseteq \Sigma^*$ we can define the following equivalence relation

$$u \sim_L v \text{ iff } \forall w \in \Sigma^* \quad u \cdot w \in L \iff v \cdot w \in L$$

- Two prefixes are equivalent if they "accept" the same suffixes

Regular Sets and their Syntactic Congruences

- With every $L \subseteq \Sigma^*$ we can define the following equivalence relation

$$u \sim_L v \text{ iff } \forall w \in \Sigma^* \;\; u \cdot w \in L \iff v \cdot w \in L$$

- Two prefixes are equivalent if they "accept" the same suffixes
- This relation is a right-congruence with respect to concatenation: $u \sim v$ implies $u \cdot w \sim v \cdot w$ for all $u, v, w \in \Sigma^*$

Regular Sets and their Syntactic Congruences

- With every $L \subseteq \Sigma^*$ we can define the following equivalence relation

$$u \sim_L v \text{ iff } \forall w \in \Sigma^* \quad u \cdot w \in L \iff v \cdot w \in L$$

- Two prefixes are equivalent if they "accept" the same suffixes
- This relation is a right-congruence with respect to concatenation: $u \sim v$ implies $u \cdot w \sim v \cdot w$ for all $u, v, w \in \Sigma^*$
- Myhill-Nerode theorem: a language L is accepted by a finite automaton iff $\sim_L$ has finitely many congruence classes
- This relation is sometimes called the syntactic congruence associated with L

The minimal Automaton

- Let $\Sigma^*/\sim$ be the quotient of Σ^* by $\sim$, that is the set of its equivalence classes and let $[u]$ denote the equivalence class of u
- The minimal automaton for L is $\mathcal{A}_L = (\Sigma, Q, q_0, \delta, F)$ where
  - The states are the $\sim$-classes: $Q = \Sigma^*/\sim$
  - Ther initial state is the class of the empty word: $q_0 = [\varepsilon]$
  - Transition function: $\delta([u], a) = [u \cdot a]$
  - Accepting states are those that accept the empty word: $F = \{[u] : u \cdot \varepsilon \in L\}$

The minimal Automaton

- Let $\Sigma^*/\sim$ be the quotient of Σ^* by $\sim$, that is the set of its equivalence classes and let $[u]$ denote the equivalence class of u
- The minimal automaton for L is $\mathcal{A}_L = (\Sigma, Q, q_0, \delta, F)$ where
  - The states are the $\sim$-classes: $Q = \Sigma^*/\sim$
  - Ther initial state is the class of the empty word: $q_0 = [\varepsilon]$
  - Transition function: $\delta([u], a) = [u \cdot a]$
  - Accepting states are those that accept the empty word: $F = \{[u] : u \cdot \varepsilon \in L\}$
- This is canonical representation of L based on its I/O semantics
- $\mathcal{A}_L$ is homomorphic to any other automaton accepting L

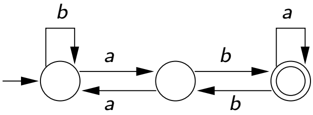
Observation Tables (Gold 1972)

- Given a language L, imagine an infinite two-dimensional table
- The rows of the table are indexed by all elements of Σ^*
- The columns of the table are indexed by all elements of Σ^*
- Each entry u, v in the table indicates whether $u \cdot v \in L$ (whether after reading prefix u we accept v)

Observation Tables (Gold 1972)

- Given a language L, imagine an infinite two-dimensional table
- The rows of the table are indexed by all elements of Σ^*
- The columns of the table are indexed by all elements of Σ^*
- Each entry u, v in the table indicates whether $u \cdot v \in L$ (whether after reading prefix u we accept v)
- For finite automata, according to Myhill-Nerode, there will be only finitely-many distinct rows (and columns)
- It is sufficient to use tables over $\Sigma^n \times \Sigma^n$

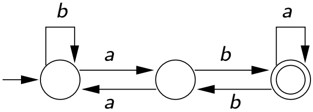
Example



	ε	a	b	aa	ab	ba	bb	$\cdots$
ε	$-$	$-$	$-$	$-$	$+$	$-$	$-$	$\cdots$
a	$-$	$-$	$+$	$-$	$-$	$+$	$-$	$\cdots$
b	$-$	$-$	$-$	$-$	$+$	$-$	$-$	$\cdots$
aa	$-$	$-$	$-$	$-$	$+$	$-$	$-$	$\cdots$
ab	$+$	$+$	$-$	$+$	$-$	$-$	$+$	$\cdots$
ba	$-$	$-$	$+$	$-$	$-$	$+$	$-$	$\cdots$
bb	$-$	$-$	$-$	$-$	$+$	$-$	$-$	$\cdots$
$\cdots$								
aba	$+$	$+$	$-$	$+$	$-$	$-$	$+$	$\cdots$
abb	$-$	$-$	$+$	$-$	$-$	$+$	$-$	$\cdots$
$\cdots$								

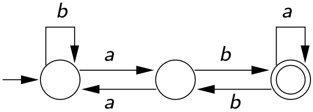
$$\varepsilon \sim b \sim aa \qquad a \sim ba \sim abb \qquad ab \sim aba$$

A Sufficient Sample to Characterize the Automaton



		E		
		ε	a	b
S	ε	$-$	$-$	$-$
	a	$-$	$-$	$+$
	ab	$+$	$+$	$-$
$S \cdot \Sigma - S$	b	$-$	$-$	$-$
	aa	$-$	$-$	$-$
	aba	$+$	$+$	$-$
	abb	$-$	$-$	$+$

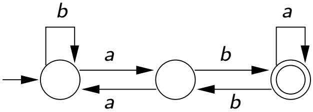
A Sufficient Sample to Characterize the Automaton



		E		
		ε	a	b
S	ε	$-$	$-$	$-$
	a	$-$	$-$	$+$
	ab	$+$	$+$	$-$
$S \cdot \Sigma - S$	b	$-$	$-$	$-$
	aa	$-$	$-$	$-$
	aba	$+$	$+$	$-$
	abb	$-$	$-$	$+$

- The states of the canonical automaton are $S = \{[\varepsilon], [a]$ and $[ab]\}$

A Sufficient Sample to Characterize the Automaton



		E		
		ε	a	b
S	ε	$-$	$-$	$-$
	a	$-$	$-$	$+$
	ab	$+$	$+$	$-$
$S \cdot \Sigma - S$	b	$-$	$-$	$-$
	aa	$-$	$-$	$-$
	aba	$+$	$+$	$-$
	abb	$-$	$-$	$+$

- The states of the canonical automaton are $S = \{[\varepsilon], [a]$ and $[ab]\}$
- The words/paths correspond to a spanning tree
- Elements of $S \cdot \Sigma - S$ correspond to cross- and back-edges in the spanning tree

Angluin's L^* Algorithm

- An incremental algorithm to construct the table based on two sources of information:
- Membership query $Member(u)$? where the learner asks whether $u \in L$
- Equivalence query $Equiv(\mathcal{A})$ where the learner asks whether automaton $\mathcal{A}$ is the (minimal) automaton for L
- The answer is either "yes" or a counter-example

Angluin's L^* Algorithm

- An incremental algorithm to construct the table based on two sources of information:
- Membership query $\mathit{Member}(u)$? where the learner asks whether $u \in L$
- Equivalence query $\mathit{Equiv}(\mathcal{A})$ where the learner asks whether automaton $\mathcal{A}$ is the (minimal) automaton for L
- The answer is either “yes” or a counter-example
- The learner asks membership queries until it can build an automaton

Angluin's L^* Algorithm

- An incremental algorithm to construct the table based on two sources of information:
- Membership query $\mathit{Member}(u)$? where the learner asks whether $u \in L$
- Equivalence query $\mathit{Equiv}(\mathcal{A})$ where the learner asks whether automaton $\mathcal{A}$ is the (minimal) automaton for L
- The answer is either "yes" or a counter-example
- The learner asks membership queries until it can build an automaton
- Then it asks an equivalence query and if there is a counter-example it adds its suffixes to the columns, thus discovering new states and so on

Angluin's L^* Algorithm

- An incremental algorithm to construct the table based on two sources of information:
- Membership query $\mathit{Member}(u)$? where the learner asks whether $u \in L$
- Equivalence query $\mathit{Equiv}(\mathcal{A})$ where the learner asks whether automaton $\mathcal{A}$ is the (minimal) automaton for L
- The answer is either "yes" or a counter-example
- The learner asks membership queries until it can build an automaton
- Then it asks an equivalence query and if there is a counter-example it adds its suffixes to the columns, thus discovering new states and so on
- Polynomial in the number of states

ω-Languages

- Let Σ^ω be the set of all infinite sequences over Σ
- An ω-language is a subset $L \subseteq \Sigma^\omega$
- The ω-regular sets can be written as a finite union of sets of the form $U \cdot V^\omega$ with U and V finitary regular sets
- Every non-empty ω-regular set contains an ultimately-periodic sequence of the form $u \cdot v^\omega$

Acceptance of ω-Languages by ω-Automata

- Consider a deterministic automaton (Σ, Q, δ, q_0)
- When an infinite word u is read by the automaton it induces an infinite run, an infinite sequence of states
- This run is summarized by $Inf(u)$, the set of states visited infinitely-often by the run

Acceptance of ω-Languages by ω-Automata

- Consider a deterministic automaton (Σ, Q, δ, q_0)
- When an infinite word u is read by the automaton it induces an infinite run, an infinite sequence of states
- This run is summarized by $Inf(u)$, the set of states visited infinitely-often by the run
- Muller acceptance condition: a set of subsets $\mathcal{F} \subseteq 2^Q$
- An infinite word u is accepted if $Inf(u) = F \in \mathcal{F}$

Subclasses of ω-Regular Sets

- If we restrict the structure of the accepting subsets $\mathcal{F}$ we obtain interesting subclasses of languages
- For example, the class **B** of languages accepted by deterministic Buchi automata
- Here we define a set F of accepting states and u is accepted if $\mathit{Inf}(u) \cap F \neq \emptyset$
- This amounts to saying that $\mathcal{F}$ consists of all elements of 2^Q that contain elements of F ($\mathcal{F}$ is upward closed)

Subclasses of ω-Regular Sets

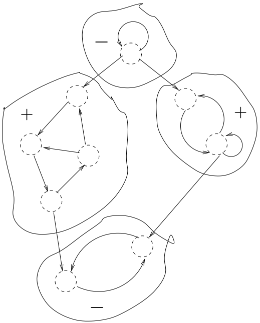
- If we restrict the structure of the accepting subsets $\mathcal{F}$ we obtain interesting subclasses of languages
- For example, the class **B** of languages accepted by deterministic Buchi automata
- Here we define a set F of accepting states and u is accepted if $Inf(u) \cap F \neq \emptyset$
- This amounts to saying that $\mathcal{F}$ consists of all elements of 2^Q that contain elements of F ($\mathcal{F}$ is upward closed)
- An infinite word u is in the complement $\bar{L}$ if $Inf(u) \cap F = \emptyset$ or equivalently $Inf(u) \subseteq Q - F$
- This is called co-Buchi condition and the class is denoted by $\bar{\mathbf{B}}$

The Class $\mathbf{B} \cap \bar{\mathbf{B}}$

- Languages that belong to both classes can be accepted by automata whose accepting set $\mathcal{F}$ admits a special structure
- In such automata, all cycles that belong to the same SCC are either accepting or rejecting

The Class $\mathbf{B} \cap \bar{\mathbf{B}}$

- Languages that belong to both classes can be accepted by automata whose accepting set $\mathcal{F}$ admits a special structure
- In such automata, all cycles that belong to the same SCC are either accepting or rejecting



- $Inf(u) \cap F \neq \emptyset$ iff $Inf(u) \cap Q - F = \emptyset$

Learning ω-Regular Sets

- First problem: how do you present examples which are infinite sequences?
- Solution: use ultimately-periodic words $u \cdot v^{\omega}$

Learning ω-Regular Sets

- First problem: how do you present examples which are infinite sequences?
- Solution: use ultimately-periodic words $u \cdot v^\omega$
- My first lemma in life: if $L \neq L'$ then there is $\alpha = u \cdot v^\omega$ that distinguishes between L and L'

Learning ω-Regular Sets

- First problem: how do you present examples which are infinite sequences?
- Solution: use ultimately-periodic words $u \cdot v^\omega$
- My first lemma in life: if $L \neq L'$ then there is $\alpha = u \cdot v^\omega$ that distinguishes between L and L'
- So now we can think of building tables where rows are words and columns are (ultimately-periodic) ω-words and entries tell us whether $u \cdot \alpha \in L$
- But it is not that simple

The Problem

- Consider the language $L = (0+1)^* \cdot 1^\omega$
- The observation table for this language looks like this

	0^ω	1^ω	$0 \cdot 1^\omega$	$1 \cdot 0^\omega$	$(01)^\omega$
ε	$-$	$+$	$+$	$-$	$-$
0	$-$	$+$	$+$	$-$	$-$
1	$-$	$+$	$+$	$-$	$-$

The Problem

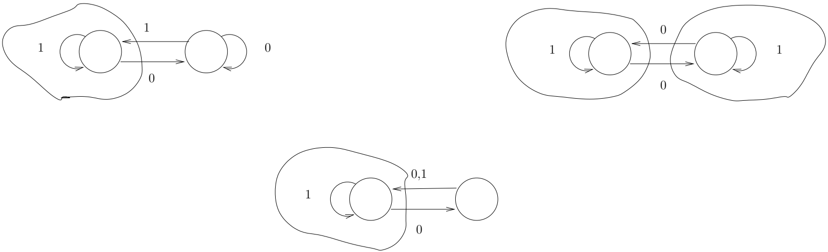
- Consider the language $L = (0+1)^* \cdot 1^\omega$
- The observation table for this language looks like this

	0^ω	1^ω	$0 \cdot 1^\omega$	$1 \cdot 0^\omega$	$(01)^\omega$
ε	$-$	$+$	$+$	$-$	$-$
0	$-$	$+$	$+$	$-$	$-$
1	$-$	$+$	$+$	$-$	$-$

- All prefixes "accept" the same language
- The Nerode congruence corresponds to a one-state automaton that, obviously, cannot accept L
- Already observed by Trakhtenbrot: in general ω-languages cannot be recognized by an automaton isomorphic to their Nerode congruence

No Canonical Minimal Automaton

- The language $L = (0+1)^* \cdot 1^\omega$ can be accepted by various 2-state automata, not related by homomorphism



Partial Solution

- Result by Staiger: languages in $\mathbf{B} \cap \bar{\mathbf{B}}$ can be recognized by their Nerode congruence

Partial Solution

- Result by Staiger: languages in $\mathbf{B} \cap \bar{\mathbf{B}}$ can be recognized by their Nerode congruence
- General culture: if we consider Cantor topology on infinite sequences
- The class $\mathbf{B} \cap \bar{\mathbf{B}}$ correspond to the class $F_\sigma \cap G_\delta$ in the Borel hierarchy
- Such sets can written as
  - Countable unions of closed sets
  - Countable intersections of open sets

Partial Solution

- Result by Staiger: languages in $\mathbf{B} \cap \bar{\mathbf{B}}$ can be recognized by their Nerode congruence
- General culture: if we consider Cantor topology on infinite sequences
- The class $\mathbf{B} \cap \bar{\mathbf{B}}$ correspond to the class $F_\sigma \cap G_\delta$ in the Borel hierarchy
- Such sets can written as
  - Countable unions of closed sets
  - Countable intersections of open sets
- We adapt Angluin's algorithm to this class

Algorithm L^ω: Sketch

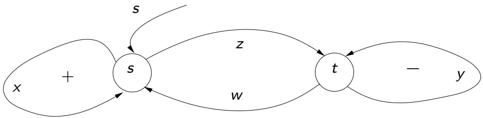
- Two phases:
  - Ask queries until you can build a transition graph for the Nerode congruence (similar to L^*)
  - Try to define a $\mathbf{B} \cap \bar{\mathbf{B}}$ acceptance condition

Algorithm L^ω: Sketch

- Two phases:
  - Ask queries until you can build a transition graph for the Nerode congruence (similar to L^*)
  - Try to define a $\mathbf{B} \cap \bar{\mathbf{B}}$ acceptance condition
- In finitary languages acceptance status for a state is determined according to whether it accepts the empty word
- For ω-languages not all cycles in the automaton are exercised infinitely-often by the sample

Algorithm L^ω: Sketch

- Two phases:
  - Ask queries until you can build a transition graph for the Nerode congruence (similar to L^*)
  - Try to define a $\mathbf{B} \cap \bar{\mathbf{B}}$ acceptance condition
- In finitary languages acceptance status for a state is determined according to whether it accepts the empty word
- For ω-languages not all cycles in the automaton are exercised infinitely-often by the sample
- We try to mark SCCs as accepting or rejecting in a way consistet with the sample, but we may have a conflict: $s \cdot x^\omega \in L$ and $s \cdot z \cdot y^\omega \notin L$. This requires more queries



Example: Learn $L = (01)^*(10)^\omega$

- Initial table is trivial, we conjecture $L = \emptyset$

	0^ω	1^ω
ε	$-$	$-$
0	$-$	$-$
1	$-$	$-$

Example: Learn $L = (01)^*(10)^\omega$

- Initial table is trivial, we conjecture $L = \emptyset$

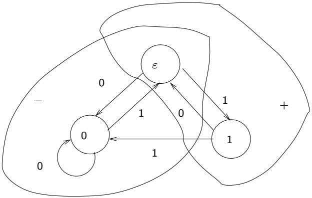
	0^ω	1^ω
ε	$-$	$-$
0	$-$	$-$
1	$-$	$-$

- We get a positive counter example $+(10)^\omega$
- We add the suffixes $(01)^\omega$ and $(10)^\omega$ to the columns and discover states 0 and 1

	0^ω	1^ω	$(01)^\omega$	$(10)^\omega$
ε	$-$	$-$	$-$	$+$
0	$-$	$-$	$-$	$-$
1	$-$	$-$	$+$	$-$
00	$-$	$-$	$-$	$-$
01	$-$	$-$	$-$	$+$
10	$-$	$-$	$-$	$+$
11	$-$	$-$	$-$	$-$

Example: Learn $L = (01)^*(10)^\omega$

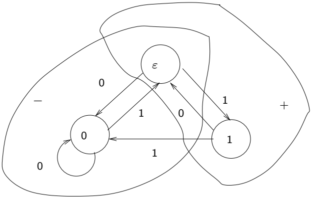
	0^ω	1^ω	$(01)^\omega$	$(10)^\omega$
ε	$-$	$-$	$-$	$+$
0	$-$	$-$	$-$	$-$
1	$-$	$-$	$+$	$-$
00	$-$	$-$	$-$	$-$
01	$-$	$-$	$-$	$+$
10	$-$	$-$	$-$	$+$
11	$-$	$-$	$-$	$-$



- The transition graph cannot be marked consistently for acceptance because $(10)^\omega \in L$ and $(01)^\omega \notin L$

Example: Learn $L = (01)^*(10)^\omega$

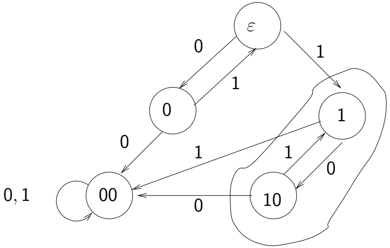
	0^ω	1^ω	$(01)^\omega$	$(10)^\omega$
ε	$-$	$-$	$-$	$+$
0	$-$	$-$	$-$	$-$
1	$-$	$-$	$+$	$-$
00	$-$	$-$	$-$	$-$
01	$-$	$-$	$-$	$+$
10	$-$	$-$	$-$	$+$
11	$-$	$-$	$-$	$-$



- The transition graph cannot be marked consistently for acceptance because $(10)^\omega \in L$ and $(01)^\omega \notin L$
- The conflict detection procedure returns the word $01(10)^\omega$ which is added together with its suffix $1(10)^\omega$ to E leading to the discovery of 2 additional states

Example: Learn $L = (01)^*(10)^\omega$

	0^ω	1^ω	$(01)^\omega$	$(10)^\omega$	$1(10)^\omega$	$01(10)^\omega$
λ	−	−	−	+	−	+
0	−	−	−	−	+	−
1	−	−	+	−	−	−
00	−	−	−	−	−	−
10	−	−	−	+	−	−
01	−	−	−	+	−	+
11	−	−	−	−	−	−
000	−	−	−	−	−	−
001	−	−	−	−	−	−
100	−	−	−	−	−	−
101	−	−	+	−	−	−



- The final table defines an automaton whose three maximal SCCs can be marked uniformly as accepting of rejecting
- This is the minimal automaton for L

Conclusions and Perspectives

- We extended learning to a subclass of ω-regular sets

Conclusions and Perspectives

- We extended learning to a subclass of ω-regular sets
- States in ω-automata have an additional "infinitary" role
- A more refined (two-sided) congruence relation was suggested by Arnold as a canonical object associated with an ω-language:

$$u \sim_L v \text{ iff } \forall x, y, z \in \Sigma^* \begin{cases} (xuyz^\omega \in L \iff xvyz^\omega \in L) \wedge \\ (x(yuz)^\omega \in L \iff x(yvz)^\omega \in L) \end{cases}$$

Conclusions and Perspectives

- We extended learning to a subclass of ω-regular sets
- States in ω-automata have an additional "infinitary" role
- A more refined (two-sided) congruence relation was suggested by Arnold as a canonical object associated with an ω-language:

$$u \sim_L v \text{ iff } \forall x, y, z \in \Sigma^* \begin{cases} (xuyz^\omega \in L \iff xvyz^\omega \in L) \wedge \\ (x(yuz)^\omega \in L \iff x(yvz)^\omega \in L) \end{cases}$$

- In [Maler Staiger 97] we proposed a smaller object, a family of right-congruences, which can, in principle, be used for learning using 3-dimensional observation tables