Learning Regular Sets

Meenakshi D'Souza

IIIT-Bangalore.

Term II 2022-'23.

Automata and Learning

- Finite state automata constitute a robust class of models in theory of computation.
- Learning was used in the context of finite state automata in the late 1980s.
- Learning regular languages helps in developing algorithms for learning invariants for programs and software design and verifying them.

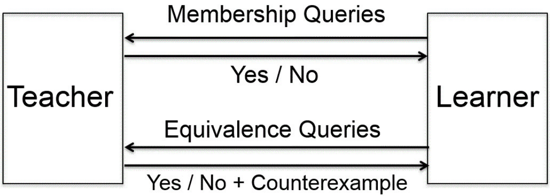
Dana Angluin

Learning Regular Sets from Queries and Counterexamples, in *Information & Computation*, 1987.

Angluin's L^* algorithm

Teacher has a regular language U in mind.
The Learner can ask two types of queries:

- Is a given string w in U? Teacher answers "Yes" or "No".
- Does a given DFA $\mathcal{A}$ accept the language U? Teacher answers "Yes" or gives a counterexample x.



Angluin's algorithm for the Learner finds the canonical DFA for U, in a number of steps polynomial in the number of states of the canonical DFA for U and the length of the longest counterexample returned by the teacher.

Angluin's Algorithm by Example

Suppose the Teacher has in mind the language

$U = \{w \in \{a,b\}^* \mid$ number of a's is even and number of b's is even$\}$

The Learner asks the Teacher if ϵ, a, and b belong to U, and obtains the following Observation Table:

		ϵ
S	ϵ	1
$S.\{a,b\}$	a	0
	b	0

The set of strings S represents the states of the automaton constructed by the Learner.
Entry (s,e) of the table represents the fact that from state s the automaton accepts/rejects the string e.

Angluin's Algorithm by Example

Suppose the Teacher has in mind the language

$U = \{w \in \{a, b\}^* \mid$ number of a's is even and number of b's is even$\}$

The Learner asks the Teacher if ϵ, a, and b belong to U, and obtains the following Observation Table:

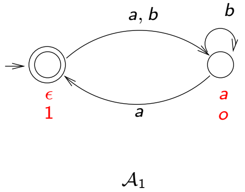
		ϵ
S	ϵ	1
$S.\{a, b\}$	a	0
	b	0

The set of strings S represents the states of the automaton constructed by the Learner.
Entry (s, e) of the table represents the fact that from state s the automaton accepts/rejects the string e.
This table is not "closed" as there are no states (or "rows") corresponding to $\epsilon \cdot a$ and $\epsilon \cdot b$.

Angluin's Algorithm by Example: 2

Learner closes table by adding string a to S, and asking membership queries for aa and ab.
He now gets the observation table:

		ϵ
S	ϵ	1
	a	0
$S.\{a, b\}$	b	0
	aa	1
	ab	0

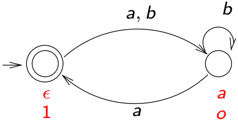


This table is closed and consistent, and represents the DFA $\mathcal{A}_1$.

Angluin's Algorithm by Example: 2

Learner closes table by adding string a to S, and asking membership queries for aa and ab.
He now gets the observation table:

		ϵ
S	ϵ	1
	a	0
$S.\{a,b\}$	b	0
	aa	1
	ab	0



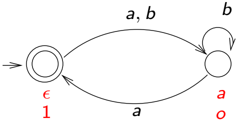
$\mathcal{A}_1$

This table is closed and consistent, and represents the DFA $\mathcal{A}_1$.
Learner now asks the Teacher if $\mathcal{A}_1$ represents the language she has in mind.

Angluin's Algorithm by Example: 2

Learner closes table by adding string a to S, and asking membership queries for aa and ab.
He now gets the observation table:

		ϵ
S	ϵ	1
	a	0
$S.\{a, b\}$	b	0
	aa	1
	ab	0



$\mathcal{A}_1$

This table is closed and consistent, and represents the DFA $\mathcal{A}_1$. Learner now asks the Teacher if $\mathcal{A}_1$ represents the language she has in mind. Teacher replies with counterexample *bb* which is in U but is not accepted by $\mathcal{A}_1$.

Angluin's Algorithm by Example: 3

Learner adds *bb* and its prefixes to his set S, makes membership queries for *ba*, *bba*, and *bbb* to obtain the observation table:

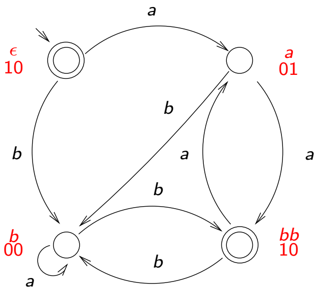
		ϵ
S	ϵ	1
	a	0
	b	0
	bb	1
$S.\{a, b\}$	aa	1
	ab	0
	ba	0
	bba	0
	bbb	0

This table is closed but not consistent. The rows for a and b are identical, but aa and ba have different rows.

Angluin's Algorithm by Example: 4

Learner adds $\epsilon \cdot a$ (that is, a) and its suffixes to the set E, and makes membership queries to obtain the observation table:

		ϵ	a
S	ϵ	1	0
	a	0	1
	b	0	0
	bb	1	0
$S.\{a,b\}$	aa	1	0
	ab	0	0
	ba	0	0
	bba	0	1
	bbb	0	0



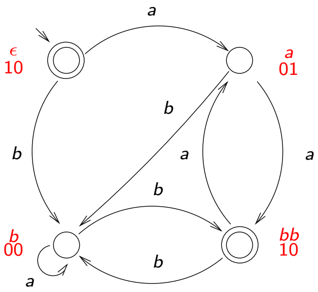
$\mathcal{A}_2$

This table is closed and consistent. So Learner conjectures the automaton $\mathcal{A}_2$.

Angluin's Algorithm by Example: 4

Learner adds $\epsilon \cdot a$ (that is, a) and its suffixes to the set E, and makes membership queries to obtain the observation table:

		ϵ	a
S	ϵ	1	0
	a	0	1
	b	0	0
	bb	1	0
$S.\{a,b\}$	aa	1	0
	ab	0	0
	ba	0	0
	bba	0	1
	bbb	0	0



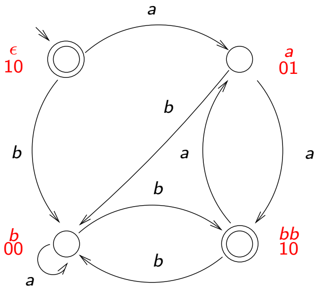
$\mathcal{A}_2$

This table is closed and consistent. So Learner conjectures the automaton $\mathcal{A}_2$. Teacher responds with counterexample abb.

Angluin's Algorithm by Example: 5

Learner adds *abb* its prefixes to S, makes membership queries to obtain the observation table:

		ϵ	a
	ϵ	1	0
S	a	0	1
	b	0	0
	ab	0	0
	bb	1	0
	abb	0	1
	aa	1	0
$S.\{a,b\}$	ba	0	0
	aba	0	0
	bba	0	1
	bbb	0	0
	$abba$	1	0
	$abbb$	0	0

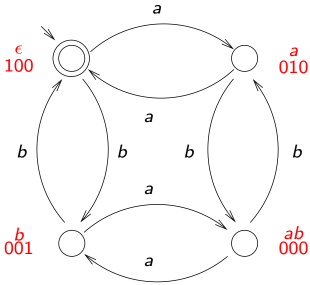


$\mathcal{A}_2$

Angluin's Algorithm by Example: 6

Learner adds b and its suffixes to E, and makes membership queries to obtain the observation table:

		ϵ	a	b
S	ϵ	1	0	0
	a	0	1	0
	b	0	0	1
	ab	0	0	0
	bb	1	0	0
	abb	0	1	0
$S.\{a,b\}$	aa	1	0	0
	ba	0	0	0
	aba	0	0	1
	bba	0	1	0
	bbb	0	0	1
	$abba$	1	0	0
	$abbb$	0	0	1



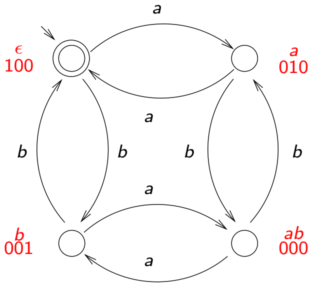
$\mathcal{A}_3$

Table is closed and consistent, so Learner conjectures DFA $\mathcal{A}_3$.

Angluin's Algorithm by Example: 6

Learner adds b and its suffixes to E, and makes membership queries to obtain the observation table:

		ϵ	a	b
	ϵ	1	0	0
S	a	0	1	0
	b	0	0	1
	ab	0	0	0
	bb	1	0	0
	abb	0	1	0
	aa	1	0	0
$S.\{a,b\}$	ba	0	0	0
	aba	0	0	1
	bba	0	1	0
	bbb	0	0	1
	$abba$	1	0	0
	$abbb$	0	0	1



$\mathcal{A}_3$

Table is closed and consistent, so Learner conjectures DFA $\mathcal{A}_3$. Teacher responds with "Yes!".

Minimally adequate teacher

A Minimally adequate teacher is supposed to answer correctly two types of queries from the learner on the unknown set U:

- Membership query: Given a string t, answer *yes* or *no* depending on whether t is a member of U or not.
- Conjecture: Given a regular set S, the answer is *yes* if $S = U$ and if not, a string t in the symmetric difference of S and the set U.

Observation table

Let A be a fixed, finite alphabet.
Observation table (S, E, T) has three components:

- A non-empty finite prefix-closed set S of strings,
- A non-empty finite suffix-closed set E of strings,
- A finite function $T : ((S \cup S \cdot A) \times E) \to \{0, 1\}$.

$T(u) = 1$ iff u belongs to the unknown regular set U.
The observation table is used to finally build the finite state automaton for U.

Closed, consistent observation tables

- An observation table is closed if for each $t \in S \cdot A$, $\exists s \in S$, $\text{row}(t) = \text{row}(s)$.
- An observation table is consistent if whenever s_1 and s_2 are in S such that $\text{row}(s_1) = \text{row}(s_2)$, then, $\forall a \in A$, $\text{row}(s_1 \cdot a) = \text{row}(s_2 \cdot a)$.

Observation tables and DFAs

Closed, consistent observation tables (S, E, T) are used to build the corresponding DFA for the unknown regular language U.
The DFA is given by $M(S, E, T) = (Q, A, \delta, q_0, F)$ where

- Set of states $Q = \{\mathrm{row}(s) \mid \ s \in S\}$,
- $q_0 = \mathrm{row}(\epsilon)$,
- $F = \{\mathrm{row}(s) \mid \ s \in S, \ T(s) = 1\}$, and
- $\delta(\mathrm{row}(s), a) = \mathrm{row}(s \cdot a)$.

M is well-defined.

Correctness of the acceptor

Theorem: If (S, E, T) is a closed, consistent observation table, then the acceptor $M(S, E, T)$ is consistent with the function T. Any other acceptor consistent with T but inequivalent to $M(S, E, T)$ must have more states.

Learner L^*

```
Initialize S and E to empty string.
Ask membership queries for empty string and each letter in alphabet.
Construct the initial observation table (S,E,T).
Repeat:
  While (S,E,T) is not closed or not consistent:
    If (S,E,T) is not consistent:
      Find s1 and s2 in S, a in A and e in E such that
      row(s1) = row(s2) and T(s1.a.e) is not equal to T(s2.a.e)
        add a.e to E, extend T to (S U S.A).E using membership queries
    If (S,E,T) is not closed:
      Find s1 in S and a in A such that
      row(s1.a) is different from row(s) for all s in S,
        add s1.a to S, extend T to (S U S.A).E using membership queries
  Once (S,E,T) is closed and consistent, let M = M(S,E,T)
  Make the conjecture M.
  If Teacher replies with a counter example t, then
    add t and all its prefixes to S
    and extend T to (S U S.A).E using membership queries
Until the Teacher replies yes to the conjecture M.
Halt and output M.
```

Automata learning for invariants

- Learns the smallest automaton that satisfies given constraints.
- This forces a kind of "generalization".
- Does it efficiently in polynomial time in smallest automaton that satisfies the given constraints.

Correctness of L^*

- L^* is correct: If the Teacher is minimally adequate then if L^* ever terminates its output is clearly an acceptor for the unknown regular set U being presented by the Teacher.
- L^* terminates:
  Lemma: Let (S, E, T) be an observation table. Let n denote the number of different values of row(s) for $s \in S$. Any acceptor consistent with T must have at least n states.

L^*: Final theorem

Theorem: Given any minimally adequate Teacher presenting an unknown regular set U, the Learner L^* eventually terminates and outputs an acceptor isomorphic to the minimal DFA accepting U. If n is the number of states of the minimum DFA accepting U and m is an upper bound on the length of any counterexample provided by the Teacher, then the total running time of L^* is bounded by a polynomial in m and n.

Conclusion

- We learnt the basics of finite state automata and a learning algorithm for learning regular languages.
- This algorithm forms the basis for algorithms for learning invariants for programs.