Exactly Learning Regular Languages Using Membership and Equivalence Queries

1 Angluin's L^* Algorithm

Introduction

In this lecture we give an exact learning procedure for the class of regular languages, using the representation class of deterministic finite automata. Suppose that the target is a regular language L over an alphabet Σ. We assume that Σ is known to the Learner; moreover we suppose that the learner has access to an oracle (called the teacher) that can answer the following two types of queries:

- **Membership queries.** In a membership query the learner selects a word $w \in \Sigma^*$ and the teacher gives the answer whether or not $w \in L$.
- **Equivalence queries.** In an equivalence query the learner selects a hypothesis automaton $\mathcal{H}$, and the teacher answers whether or not L is the language of $\mathcal{H}$. If yes, then the algorithm terminates. If no, then the teacher gives a counterexample, i.e., a word in which L differs from the language of $\mathcal{H}$.

In this setting we present a learning procedure, due to Dana Angluin, called the L^* algorithm. This algorithm is guaranteed to learn the target language using a number of queries that is polynomial in:

- the number of states of a minimal deterministic automaton representing the target language;
- the size of the largest counterexample returned by the teacher.

In fact it will turn out that if the teacher always returns a counterexample of minimal length then the total number of queries is polynomial in the size of the minimal automaton for the target language.

Deterministic Finite Automata.

Recall that a deterministic finite automaton (DFA) is a tuple $(\Sigma, Q, q_0, \delta, F)$, where Σ is a finite alphabet, Q is is a finite set of states, $q_0 \in Q$ is the initial state, $\delta : Q \times \Sigma \to Q$ is the transition function, and $F \subseteq Q$ is the set of final (or accepting) states. We extend δ to a function $\delta : Q \times \Sigma^* \to Q$ by $\delta(q, \varepsilon) = q$ and $\delta(q, wa) = \delta(\delta(q, w), a)$ for all $a \in \Sigma^*$ and $w \in \Sigma^*$. The language accepted by $\mathcal{A}$ is $\{w \in \Sigma^* : \delta(q_0, w) \in F\}$.

Access Words and Test Words.

Suppose that the target language is $L \subseteq \Sigma^*$. At each step of the algorithm, the learner maintains:

- A set $Q \subseteq \Sigma^*$ of *access words*, with $\varepsilon \in Q$.
- A set $T \subseteq \Sigma^*$ of *test words*.

Given a set T of test words, we say that $v, w \in \Sigma^*$ are T-*equivalent*, denoted $v \equiv_T w$, if

$$vu \in L \text{ iff } wu \in L \quad \text{for all } u \in T\,.$$

Intuitively, access words are used by the learner to identify the different Brzozowski derivatives of the target language L, while test words are used to distinguish different derivatives.

We define the following two properties of the sets Q and T:

- **Separability**: no two distinct words in Q are T-equivalent.
- **Closedness**: for every $q \in Q$ and $a \in \Sigma$, there is some $q' \in Q$ such that $qa \equiv_T q'$.

If (Q, T) is separable and closed then we can define a *hypothesis automaton* $\mathcal{H}$ as follows. The set of states of $\mathcal{H}$ is Q, with the empty word ε being the initial state. When $\mathcal{H}$ is in state $q \in Q$ and reads a letter $a \in \Sigma$, then it makes a transition to the unique state $q' \in Q$ such that $qa \equiv_T q'$. (Such a state exists by closedness and is unique by separability.) The accepting states of $\mathcal{H}$ are those $q \in Q$ that lie in the target language L.

The learning procedure is based on the following three propositions:

Proposition 1. *If (Q, T) is separable then $|Q|$ is at most the number of states of a minimal DFA for L.*

Proof. Let $\mathcal{A}$ be a DFA for the language L and denote by q_0 and δ the initial state and transition function of $\mathcal{A}$. Clearly, any two words $u, v \in \Sigma^*$ are T-equivalent if $\delta(q_0, u) = \delta(q_0, v)$, i.e., $\mathcal{A}$ ends in the same state after reading u and v respectively. But then separability of Q entails that $|Q|$ is at most the number of states of $\mathcal{A}$. $\square$

Proposition 2. *If (Q, T) is separable but not closed, then using membership queries one can find $q \in \Sigma^* \setminus Q$ such that $(Q \cup \{q\}, T)$ remains separable.*

Proof. Since (Q, T) is not closed, there exists $q \in Q$ and $a \in \Sigma$ are such that qa is not T-equivalent to any $q' \in Q$. Using membership queries we can find such a q and a. We then add qa to Q. This maintains separability by construction. $\square$

Proposition 3. *Suppose that (Q, T) is separable and closed and let $\mathcal{H}$ be the hypothesis automaton. Given a counterexample $w = w_1 \dots w_n$ to $\mathcal{H}$, using $\log |w|$ membership queries, one can find $q \in \Sigma^* \setminus Q$ and $t \in \Sigma^*$ such that $(Q \cup \{q\}, T \cup \{t\})$ is separable.*

Proof. Let $q_0 = \varepsilon$ be the initial state of $\mathcal{H}$ and δ the transition function of $\mathcal{H}$. For $i = 1, \dots, n$, define $q_i = \delta(q_0, w_1 \dots w_i)$ to be the state reached by $\mathcal{H}$ after reading the prefix $w_1 \dots w_i$ of w.

Writing χ_L for the characteristic function of the target language L, we say that state q_i is *correct* if $\chi_L(q_i w_{i+1} \dots w_n) = \chi_L(w)$. Note that correctness of q_i can be checked with a membership query. Now state $q_0 = \varepsilon$ is obviously correct, while state q_n is not correct since w is a counterexample and hence $\chi_L(q_n) \neq \chi_L(w)$ by definition of the set of accepting states of $\mathcal{H}$. Thus, using binary search, one can find i such that q_{i-1} is correct and q_i is not correct, that is,

$$\chi_L(q_{i-1} w_i \dots w_n) \neq \chi_L(q_i w_{i+1} \dots w_n) .$$

Now let $Q' = Q \cup \{q_{i-1} w_i\}$ and $T' = T \cup \{w_{i+1} \dots w_n\}$. By definition of the transition function of $\mathcal{H}$, q_i is the unique element of Q that is T-equivalent to $q_{i-1} w_i$. On the other hand, the test $w_{i+1} \dots w_n$ distinguishes q_i from $q_{i-1} w_i$. We conclude that $q_{i-1} w_i \notin Q$ and that (Q', T') is separable. $\square$

The Algorithm

We are now ready to describe the algorithm. Throughout any execution (Q, T) remains separable but not necessarily closed.

1. $Q := T := \{\varepsilon\}$
2. Repeatedly applying Proposition 2, enlarge Q such that (Q, T) separable and closed.
3. Compute the hypothesis automaton for (Q, T) and ask an equivalence query for it.
4. If the answer is yes, then the algorithm terminates with success.

5. If the answer is no, then apply Proposition 3 to properly expand Q and T to obtain a separable pair (Q', T').

6. Goto 2.

Theorem 1. *The representation class of deterministic finite automata is efficiently learnable using equivalence and membership queries.*

Proof. Consider a run of the L^* algorithm, given target language L over alphabet Σ. Let m be the number of states of a minimal automaton for L and let n be the length of the largest counterexample returned by the teacher.

From Proposition 1 we deduce that the number of equivalence queries is at most m, since each equivalence query leads us to expand Q with at least one element.

Associated with each equivalence query we have at most $\log n$ membership queries (Proposition 3). Thus we make at most $m \log n$ membership queries in Step 5 of the algorithm.

Each membership query in Step 2 of the algorithm is performed on a word of the form qt or qat, where $q \in Q$, $a \in \Sigma$, and $t \in T$. Since $|T| \leq |Q| = m$ on termination, the total number of such queries is at most $(|Q| + |Q||\Sigma|)|T| \leq m^2(1 + |\Sigma|)$.

Thus we have an overall polynomial bound in n, m, and $|\Sigma|$ on the number of queries. Given this, it is obvious that the running time is also polynomially bounded. $\square$

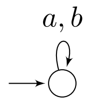
2 Examples and Applications

A Counting Language

Consider a run of Angluin's algorithm with target language

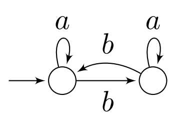
$$L = \{w \in \{a, b\}^* : \text{ the number of } b\text{'s in } w \text{ is congruent to 3 modulo 4}\}.$$

1. Initially we have $Q = T = \{\varepsilon\}$. Notice that (Q, T) is closed and separable. In particular, we have $a \equiv_T \varepsilon$ and $b \equiv_T \varepsilon$. Thus we may construct a hypothesis automaton:



This automaton has an empty language. Suppose that the learner performs an equivalence query and receives counterexample bbb. Performing Step 5 of the algorithm, we expand Q and T to obtain $Q = \{\varepsilon, b\}$ and $T = \{\varepsilon, bb\}$.

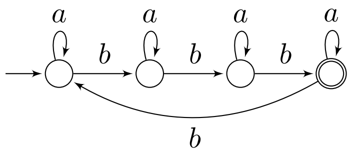
2. Again, (Q, T) is closed and separable. Thus we may construct a hypothesis automaton:



Again this automaton has empty language. Suppose that the learner performs an equivalence query and receives counterexample bbb. Performing Step 5 of the algorithm we expand Q and T to obtain $Q = \{\varepsilon, b, bb\}$ and $T = \{\varepsilon, b, bb\}$.

3. Now (Q, T) is no longer closed, since $bbb \not\equiv_T \varepsilon, b, bb$. Thus we update (Q, T) to $Q = \{\varepsilon, b, bb, bbb\}$ and $T = \{\varepsilon, b, bb\}$.

4. Now (Q, T) is closed and separable. The hypothesis automaton is



Performing an equivalence query, we see that this exactly represents the target language.

Learning Conjunctions of Linear Classifiers

Recall that a *linear classifier* is a function $f : \{0,1\}^n \to \{-1,+1\}$ of the form

$$f(x_1, \dots, x_n) = \operatorname{sign}\left(\sum_{i=1}^{n} a_i x_i + b\right)$$

for given integers $a_1, \dots, a_n, b$. The *weight* of such a classifier f is defined to be $W = \sum_{i=1}^{n} |a_i| + |b|$.

We can naturally represent a linear classifier $f : \{0,1\}^n \to \{-1,+1\}$ as the language of a DFA $\mathcal{A}$ over alphabet $\{0,1\}$, where $\mathcal{A}$ accepts a word $x_1 \dots x_n \in \{0,1\}^n$ if and only if $f(x_1, \dots, x_n) = 1$.

Exercise 1. *Show that a linear classifier $f : \{0,1\}^n \to \{-1,+1\}$ of weight W can be represented by a DFA with number of states $O(nW)$.*

Exercise 2. *Show that a conjunction of k linear classifiers, each of weight at most W, can be represented by a DFA with number of states $O((nW)^k)$.*

Proposition 4. *For each fixed k, the representation class of conjunctions of k linear classifiers is exactly learnable using the representation class of DFA with number of queries polynomial in the total weight of the target classifier.*