Exactly Learning Regular Languages with Membership and Equivalence Queries

James Worrell: 8/2/2017

1 Angluin's L^* Algorithm

Introduction

In this lecture we give an exact learning procedure for regular languages using the representation class of deterministic finite automata. Suppose that the target is a regular language L defined over an alphabet Σ. We assume that Σ is known to the Learner and 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 finishes. 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, called the L^* algorithm, due to Dana Angluin with refinements by Rivest and Schapire. 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.

Note that if the teacher always returns a counterexample of minimal length then the second item above is superfluous, i.e., the total number of queries is polynomial in the size of the minimal DFA for the target language.

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

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

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* $\mathcal{H}$, based on (Q, T). The set of states of $\mathcal{H}$ is Q, with the empty word being the initial state. When $\mathcal{H}$ is in state $q \in Q$ and reads a letter $a \in \Sigma$, then it goes to the 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. Due to separability, any two distinct words $p, q \in Q$ must lead from the initial state of $\mathcal{A}$ to distinct states of $\mathcal{A}$. (Why?) □

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. □

Proposition 3. *Suppose that (Q, T) is separable and closed and let $\mathcal{H}$ be the hypothesis automaton. Given a counterexample w 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 $w = w_1 \dots w_n$ and let $q_0 \xrightarrow{w_1} q_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} q_n$ be a run of $\mathcal{H}$ on w. Identifying $L \subseteq \Sigma^*$ with its characteristic function $\Sigma^* \to \{0, 1\}$, say that state q_i is *correct* if $L(q_i w_{i+1} \dots w_n) = L(w)$. State q_0 is correct since $q_0 = \varepsilon$, and state q_n is not correct since w is a counterexample. Using binary search one can find i such that q_{i-1} is correct and q_i is not correct, that is,

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

Now let $Q' = Q \cup \{q_{i-1} w_i\}$ and $T' = T \cup \{w_{i+1} \dots w_n\}$. Then $q_{i-1} w_i \notin Q$ and (Q', T') is separable. (Why?) □

The Algorithm

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

1. $Q := T := \{\varepsilon\}$
2. Repeatedly applying Proposition **??**, 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 **??** 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.

If (Q,T) is the state of the algorithm when it terminates then the total number of membership queries is at most

$$(|Q| + |Q||\Sigma|)|T| + |Q|\log n \leq (m + m|\Sigma|)m + m\log n$$

(since $|Q| \leq m$ by Proposition **??**). The number of equivalence queries is at most m since each equivalence query leads us to expand Q with at least one element. 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. $\square$

Exercise 1. *Argue that the set of access strings Q is always prefix closed.*

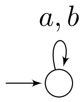
2 Examples and Applications

A Counting Language

Consider a run of Angluin's algorithm, given target language

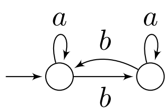
$$L = \{w \in \{a,b\}^* : \text{ the number of } b\text{'s in } w \text{ is congruent to 3 modulo 4}\} .$$

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 $T = \{\varepsilon, bb\}$.

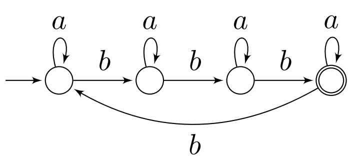
2. Again, (Q,T) is closed and separable. Thus we may construct a hypothesis automaton:

   Automaton $\mathcal{H}$ 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

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

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 classifer $f : \{0,1\}^n \to \{-1,+1\}$ 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 2. *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 3. *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 efficiently exactly learnable using the representation class of DFA.*