LAST TIME:

Exact Learning using Membership Queries (MQ) and Equivalence Queries (EQ)

Membership Query (MQ) Oracle:

Learning algorithm queries with $x \in X$, oracle responds with $c(x)$ where c is the target concept.

Equivalence Query (EQ) Oracle:

Learning algorith submits a (description of) hypothesis $h: X \to \{0,1\}$. Oracle responds with "success" if $h \equiv c$ (c is the target) OR Oracle returns $x \in X$, s.t. $h(x) \neq c(x)$ (counter example)

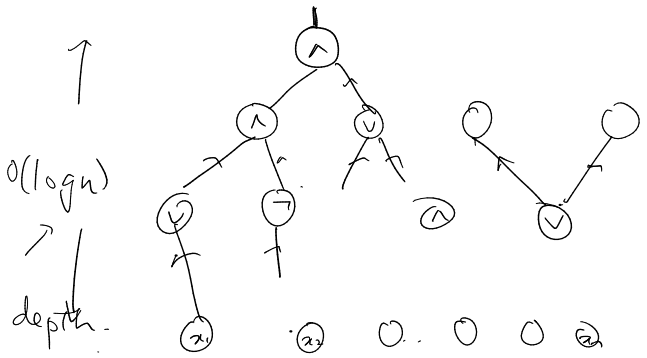
Defn: We say that the concept class C is efficiently learnable using MQ + EQs is there is a polynomial $p(\cdot,\cdot)$ and a learning algorithm L, that $\forall n \geq 1$, $\forall c \in C_n$, with access to MQ & EQ oracles for c, outputs in time $p(n, \text{size}(c))$ a hypothesis h, s.t. $h \equiv c$.

(Exercise: Show that if a concept class is exactly learnable using MQs + EQs, then it is also PAC-learnable using MQs in addition to the example oracle $EX(c, D)$. (See §4 on PS 3))

[Exercise: Show that MONOTONE-DNF formulae with at most s terms are efficiently exactly learnable using MQs + EQs in time poly(n, s).]

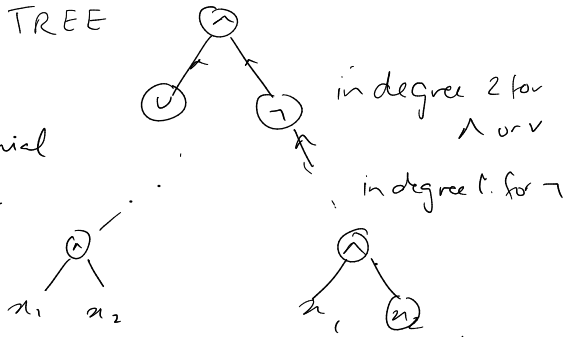
- Previously we have "seen" (almost) that the concept class C of functions representable using boolean circuits of depth at most $O(\log n)$ are not efficiently PAC learnable under the DCRA.

- Boolean circuit (DAG)



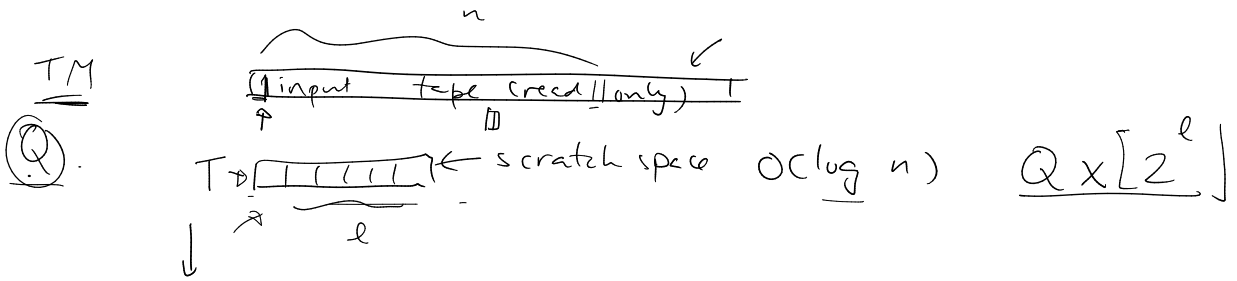
(Exercise)
Formula of size → polynomial in n

Boolean Formula (TREE)



leaves are labelled by variables

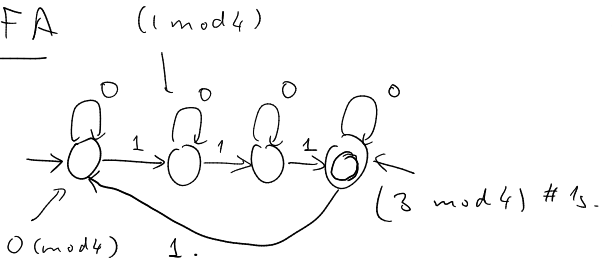
Any boolean formula of size poly(n) can be evaluated by a Turing Machine which uses space $O(\log n)$.



DFA of size at most poly(n).
distinct states of the TM $\leq n^c$

- Learning DFA of polynomial size in the PAC-model is hard under the DCRA.

DFA



L = all strings of 0's & 1's where #1s is 3 mod 4.

- # states in the smallest DFA representing the language L.
- length of the longest counterexample returned by the equivalence oracle.

$L \subseteq \Sigma^*$, where Σ is a finite alphabet, $\Sigma = \{0, 1\}$.

$Q \subseteq \Sigma^*$ is a set of "access" strings, $\varepsilon \in Q$ (empty string)

$T \subseteq \Sigma^*$ is a set of "test" strings.

"Q should represent the states of the DFA that we build"

Two strings u & $w \in \Sigma^*$ are T-equivalent $u \equiv_T w$ if

$uv \in L \iff wv \in L \quad \forall v \in T$.

Properties of the pair (Q, T).

- (i) Q is separable, no two strings are T-equivalent.
- (ii) Q is closed, $\forall q \in Q, \forall a \in \Sigma, \exists q' \in Q$, s.t. $qa \equiv_T q'$.

(qa: concatenation)

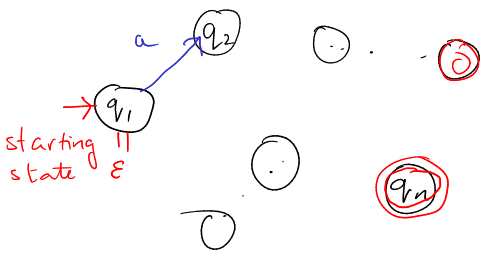
Lemma (1): If (Q, T) is separable, then $|Q|$ is at most the # states in the smallest DFA recognizing L.

Proof: Suppose not, $\exists q_1, q_2$ that take you to the same state in the DFA. Then $q_1 u$ & $q_2 u$ are in the same state $\forall u \in \Sigma^*$. This contradicts q_1 & q_2 not being T-equivalent.

Lemma (2): If (Q, T) are separable & closed then we can we can build a DFA.

Proof: Check that $\equiv_T$ is an equivalence relation.

(reflexive, symmetric & transitive)



if $q_i a \equiv_T q_j$ (uniquely defined)

State q is accepting if $q \in L$.

- (Q, T) is closed
- $\equiv_T$ equivalence relation.

Lemma ③: If (Q, T) is separable but not closed, then we can find using MQ a string q s.t. $(Q \cup \{q\}, T)$ remains separable.

Proof: $\forall q \in Q$, $\forall a \in \Sigma$, check if $\exists q' \in Q$, s.t. $qa \equiv_T q'$ ("checking")

If not then add qa to Q.

Checking can be done using at most $O(|Q|.|T|.|\Sigma|)$ membership queries.

Lemma 4: If (Q, T) are separable and closed, by the DFA, $A_{(Q,T)}$ obtained using Lemma ② is not equivalent to the target, then, given a counterexample w, using at most $|w|$ membership queries, we can find $q' \notin Q$, & $t' \notin T$ s.t. $(Q \cup \{q'\}, T \cup \{t'\})$ is separable.

Proof: Let $w = w_1 w_2 \ldots w_n$ be the counterexample $w_i \in \Sigma$,

$q_0 \xrightarrow{w_1} q_1 \xrightarrow{w_2} q_2 \cdots \xrightarrow{w_n} q_n$.

q_i is "correct" if $q_i w_{i+1} \ldots w_n \in L \Leftrightarrow w \in L$.

q_0 is "correct" (obviously as q_0 represents the empty string ε)

q_n is "incorrect". (" as $q_n \in L \Leftrightarrow w \notin L$)

$\exists i$, s.t q_{i-1} is correct but q_i is not. $(q' \notin Q)$

$q' = q_{i-1} w_i$ $\quad t' = w_{i+1} \ldots w_n$.

This allows us to grow Q and T as required.

$q_{i-1} w_i \xrightarrow{\quad} w_n \in L \Leftrightarrow w \in L$

$\|$

q_i

→ The fact that q_{i-1} is correct & q_i is not $\Rightarrow q' \notin Q$.

→ (Q', T') remains separable.

$q_{i-1} w_i \equiv_T q_i$ (before additions)

$q_{i-1} w_i t' \in L \Leftrightarrow q_i t' \notin L$.

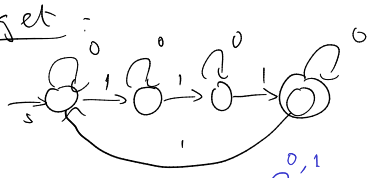
∎

Algorithm :

- $Q = \{\varepsilon\}$, $T = \{\varepsilon\}$
- while (true) {
  - → Use Lemma 3 to make (Q, T) separable & closed.
  - → Use Lemma 2 to make DFA $A_{(Q,T)}$
  - → Make EQ($A_{(Q,T)}$)
    - if success; break.
    - else $Q \leftarrow Q \cup \{q'\}$, $T \leftarrow T \cup \{t'\}$ } using Lemma (4)
    - & counter example w.

}.

Target :



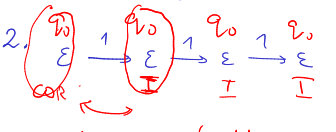
counter example

111.

1. $Q = \{\varepsilon\}$, $T = \{\varepsilon\}$.

$\varepsilon \equiv_T \varepsilon 0$
$\varepsilon \equiv_T \varepsilon 1$] checked using MQs

2.

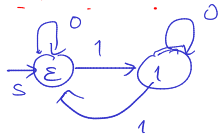


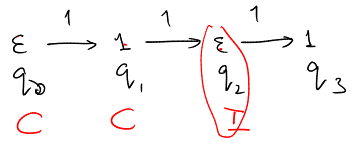
$Q = \{\varepsilon, 1\}$, $T = \{\varepsilon, 11\}$.

$\varepsilon 0 \equiv_T \varepsilon$
$\varepsilon 1 \equiv_T 1$
$11 \equiv_T \varepsilon$
$10 \equiv_T 1$

111

$q' = 1$, $t' = 11$





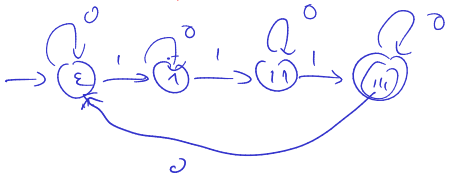
$q' = 11$
$t' = 1$

3. $Q = \{\varepsilon, 1, 11\}$ $T = \{\varepsilon, 1, 11\}$ (separable but not closed)

$111 \neq_T \varepsilon, 1, 11$

$Q = \{\varepsilon, 1, 11, 111\}$

4.



Under DCRA

Efficient PAC ≠ Efficient PAC + MQ