Fundamentals of MCS: CS - Part 2

NGUYEN T. Hoang - SID: 15M54097

Fall 2015, W832 Mon. Period 7-8

Due date: 2016/01/18

Problem

For this Part II of Fall 2015 Fundamentals of Mathematical and Computing Sciences: Computer Science class, I choose problem 3 for grading.

Q3.1. We would like to port the compiler to another stack machine whose behavior is slightly different from the original one. Although the representation of its structure remains the same (Definition prog := list instr and Definition stack := list nat), the new stack machine's interpretation of instruction is slightly different:

Q.3.1. It is not so difficult to prove Theorem 1, so why don't you prove it (without reading the referenced paper)! You can go back to the definition and consider the probability that one fixed hypothesis h is not an ϵ-approximation of a given target f_* on m example of S. (What is the randomness here for discussing the probability?) Then we can use the union bound to estimate the probability that this situation occurs on some hypothesis of $\mathcal{H}_{n,m}$.

Theorem 1. *(PAC learning is achieved by "Occam Razor")*
For any concept class $\mathcal{C}$, consider any algorithm L that yields a hypothesis consistent with a given sample. Let $\mathcal{H}_{n,m}$ be a class of hypotheses (i.e., Boolean functions) that algorithm L may yield on some sameple of size m on some target concept in $\mathcal{C}$ of size n. (Note that m is determined by algorithm L from ϵ, δ, n).
Let $M(n, m)$ denote the number of hypotheses of $\mathcal{H}_{n,m}$. For any learning parameters ϵ, δ, and for any n, if we can design the algorithm so that

$$m \geq \frac{1}{\epsilon} \ln \frac{1}{\delta} + \frac{\ln M(n, m)}{\epsilon}$$

holds, then L can be used as a PAC-learning algorithm for $\mathcal{C}$.

Answer:

Q1.1 - Regular language definition with $\approx_L$.

My proof contains two parts:

1. A language L is regular $\Rightarrow \Sigma^*$ can be divided into a finite number of equivalence classes with $\approx_L$, and
2. A language L that has Σ^* can be divided into a finite number of equivalence classes $\Rightarrow L$ is regular.

A regular language is defined as follow:

> A language is called ***regular language*** if some finite automaton recognizes it. [**?**]

Also, with some deterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$, I define notations for string collection R_i and state transition function based on string input δ^* mentioned in [**?**] as follow:

- R_i is a set of input strings, which makes the automaton M transits from q_0 to a same state q_i in Q.

$$R_i = \{u \mid u \in \Sigma^* \text{ and } \delta^*(q_0, u) = q_i \in Q\} \tag{1}$$

- $\delta^* : Q \times \Sigma^* \to Q$ is an extended transition function that takes a string and a state as inputs. The output of this function is the state of the automaton M after taking the string of input. For any $q \in Q$:

$$\begin{cases} \delta^*(q, \varepsilon) = \delta(q, \varepsilon) \\ \delta^*(q, e \cdot w) = \delta^*(\delta(q, e), w), \quad \forall e \in \Sigma \text{ and } w \in \Sigma^* \end{cases}$$

1. For the first part of the proof, I consider L, a regular language, and I will prove that there is a way to divide Σ^* into a finite set by the equivalent relation $\approx_L$. Since L is regular, there exists a deterministic automaton $M = (Q, \Sigma, \delta, q_0, F)$ accepts L. For each state in Q of the automaton M, consider a set of strings R_i as defined above. Since the number of state in M is $|Q|$, and it is a finite number, hence we have a finite number of R_i.

By definition in [**?**], the u-derivative of L is stated as follow:

$$\partial_u L = \{v : u \cdot v \in L\}$$

Intuitively, an u-derivative of L is the set of all substring of strings that start with u in L. Using the extended transition function, we have another representation of this deriviation:

$$\partial_u L = \{w : \delta^*(q_0, u \cdot w) \in F\}$$

For all set R_i, and for any $u, v \in R_i$:

$$\begin{aligned} \partial_u L &= \{w : \delta^*(\delta^*(q_0, u), w) \in F\} \\ \partial_v L &= \{z : \delta^*(\delta^*(q_0, v), z) \in F\} \end{aligned} \tag{2}$$

By the definition of R_i, we denote $\delta^*(q_0, u) = \delta^*(q_0, v) = q_i$, formula (2) can be rewritten as follow:

$$\begin{aligned} \partial_u L &= \{w : \delta^*(q_i, w) \in F\} \\ \partial_v L &= \{z : \delta^*(q_i, z) \in F\} \end{aligned} \text{, for all } u, v \in R_i.$$

Because of the determinism of the automaton M and its state transition function, we have:

$$\begin{cases} \{w : \delta^*(q_i, w) \in F\} \\ \{z : \delta^*(q_i, z) \in F\} \end{cases} \quad \Leftrightarrow \quad \{w : \delta^*(q_i, w) \in F\} \equiv \{z : \delta^*(q_i, z) \in F\}$$

Therefore:

$$\begin{aligned} \partial_u L &= \partial_v L \\ \text{hence,} \quad u &\approx_L v \quad , \forall u, v \in R_i \end{aligned} \tag{3}$$

From formula (3), we see that dividing Σ^* into a finite set $\{R_i : i = (0 \ldots |Q| - 1)\}$ is equivalent with dividing the language into a finite set by the equivalent relation $\approx_L$.

2. In the second part of this proof, I consider there is a language L, and Σ^* can be divided into a finite set R by the equivalent relation $\approx_L$. I will construct a deterministic finite automaton for recognizing the given language L. For all R_i in the finite set R:

$$u, v \in R_i \Leftrightarrow \partial_u L = \partial_v L$$

Also:

$$u \in \Sigma^* \Leftrightarrow \exists R_k \in R : u \in R_k$$

I construct the deterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ by defining each element of the 5-tuples. For each class R_i divided by the equivalent relation, I define a state q_i corresponds with it. The alphabet Σ is the symbol set of the given language L. q_0 is the state corresponds with R_0, containing an empty string ε. F is the set of all strings that have deriviation over L equals empty string ε. Formally, we have:

1. $Q = \{q_i : \text{Each } q_i \text{ corresponds with a class } R_i\}$.
2. q_0 is the start state corresponds with the class of empty string $R_0 = \{\varepsilon\}$.

3. $\forall e \in \Sigma,$ we have: $\delta(q_i, e) = q_j : \partial_{u \in R_j} L = \partial_{u \cdot e,\ u \in R_i} L$. Because there always exists a state q_j correspondings to R_j that contains the string $u \cdot e$.

4. $F = \{q_f$ corresponds to $R_f : \forall u \in R_f,\ \partial_u L = \varepsilon\}$

In conclusion, I have proved that: **1.** If the language L is regular, we can divide Σ^* into a finite set by the equivalent relation $\approx_L$, and **2.** If the language L has Σ^* can be divided into a finite set by the equivalent relation $\approx_L$, we can construct a deterministic finite automaton M that recognizes L. Therefore:

$$L \text{ is a } \textbf{regular language} \Leftrightarrow \begin{array}{l}\Sigma^* \text{ can be divided into a finite number} \\ \text{of equivalent classes by } \approx_L\end{array} \qquad \text{(Q.E.D.)}$$

Consider a non-regular language from [**?**]: $E = \{0^i 1^j\ , i > j\}$. E is a language which was proven to be non-regular in [**?**] using the Pumping lemma. Here, I will show that the number of equivalent class divided by $\approx_L$ is infinite. The string in E has 2 parts of leading 0's and trailing 1's. The number of leading 0's is i and that of trailing 1's is j. Suppose we have a non empty string $u_{i,j}$, which has i < j (Number of leading 0's larger than number of leading 1's). The derivative of such string is:

$$\begin{aligned}\partial_{u_{i,j}} E &= \{\text{Set of all-1 strings of length less than i-j}\} \\ &= \left\{ \bigcup_{k=0}^{i-j-1} 1^k \right\}\end{aligned} \tag{4}$$

Equation (4) shows that the size $|\partial_{u_{i,j}} E|$ depends on the choice of the pair (i, j). The number of set of all-1 strings is the number of values that (i-j) has. Therefore, the number of equivalent classes divided by $\approx_L$ is infinite. Another example considers the language $L = \{\{0,1\}^*\}$. Language L was also proven to be non-regular in [**?**]. Any string $u \in \{0,1\}^*$ can be followed by any string in $\{0,1\}^*$, and the concatenated string is also in L. Therefore, the number of equivalent classes is infinite.

$$\partial_u L = \{0,1\}^*\ , \forall u \in \{0,1\}^* \tag{5}$$

Generally, if a language is non-regular, it is impossible to define a DFA or NFA that recognizes the language. The automaton can recognize a non-regular language will have an infinite number of states, and the for any string pair of length k, there is no finite length string to distinguish them. Hence, the language is not divisible to a finite number of equivalent classes.

Q3.1. Proof of *"Occam Razor"* PAC-Learning

Consider the condition for an algorithm A to be called a *PAC-learning algorithm* for a given concept class $\mathcal{C}$: [**?**]

$$\begin{aligned}
&\forall \epsilon, \delta, 0 < \epsilon, \delta < 1, \forall n \geq 1, \\
&\exists m \geq 0 \text{ which is determined by } A \text{ from } \epsilon, \delta, n \\
&\forall D_*(\text{distribution over } \{+1, -1\}^n), \forall f_* \in \mathcal{C} \\
&\Pr_{S:D_*^m}\left[\begin{array}{c} A \text{ given } S \text{ yields some h satisfying} \\ \Pr_{\boldsymbol{x}:D_*}[f_*(\boldsymbol{x}) \neq h(\boldsymbol{x})] \leq \epsilon \end{array}\right] \geq 1 - \delta.
\end{aligned} \tag{6}$$

Suppose we have a hypothesis h_{bad} satisfies **Theorem 1** given to us consistenly by algorithm L, but h_{bad} is not an ϵ-estimation of a given target concept f_*.

$$\Pr_{\boldsymbol{x}:D_*}[f_*(\boldsymbol{x}) \neq h_{bad}(\boldsymbol{x})] \leq \epsilon$$

I denote event $\mathcal{X}$ is the event where h_{bad} gives consistent output with the target concept f_*, the probability of X with a random sample in D_* is:

$$\Pr[\mathcal{X}] = \Pr_{\boldsymbol{x}:D_*}[f_*(\boldsymbol{x}) = h_{bad}(\boldsymbol{x})] \geq 1 - \epsilon$$

Given m random samples from distribution D_*, the worst case can happen is $\mathcal{X}$ holds for all m samples. Therefore, the probability of this case is at most:

$$\Pr[\mathcal{X}^m] = (1 - \epsilon)^m$$

According to **Theorem 1**, the number of hypotheses of class $\mathcal{H}_{n,m}$ is $M(n, m)$. I denote $\mathcal{Y}$ as the event that the algorithm L gives us h_{bad} over all hypothesis $h_i \in \mathcal{H}_{n,m}$. The union bound gives us the probability for event $\mathcal{Y}$:

$$\begin{aligned}
\Pr[\mathcal{Y}] = \Pr\left[\bigcup_{h:\mathcal{H}_{n,m}} \mathcal{X}_h^m\right] &\leq \sum_{h:\mathcal{H}_{n,m}} \Pr[\mathcal{X}_h^m] \\
&\leq M(n, m) \times (1 - \epsilon)^m
\end{aligned} \tag{7}$$

Inequality (7) means that the chance for our algorithm L consistently gives hypotheses that are not ϵ-estimation of a given target concept f_* is bounded by $M(n, m) \times (1 - \epsilon)^m$. For this reason, we would like to bound this quantity by a confident parameter δ, and then solve the inequality for the sameple size m.

$$\begin{aligned}
& M(n, m) \times (1 - \epsilon)^m && \leq && \delta \\
\Leftrightarrow \quad & (1 - \epsilon)^m && \leq && \frac{\delta}{M(n, m)} \\
\Leftrightarrow \quad & \ln(1 - \epsilon)^m && \leq && \ln \frac{\delta}{M(n, m)} \\
\Leftrightarrow \quad & m \times \ln(1 - \epsilon) && \leq && \ln \delta - \ln M(n, m) \quad (*)
\end{aligned}$$

Here, I will prove that the inequality $\ln(1-\epsilon)+\epsilon \leq 0$ holds true for $\epsilon \in (0,1)$. Consider the function $g(x) = \ln(1-x)+x$, for $x \in (0,1)$, take the derivative of this function we have:

$$\begin{aligned} \frac{d(g(x))}{dx} &= \frac{d}{dx}(\ln(1-x)+x) \\ &= \frac{x}{x-1} \ \leq \ 0 \ \ \forall x \in (0,1) \end{aligned} \tag{8}$$

We also have:

$$g(0) = \ln(1-0)+0 = 0 \tag{9}$$

(8) and (9) show that $g(x)$ is a decreasing function over $(0,1)$ and $g(0)=0$. Therefore, $g(x) \leq 0$ for $x \in (0,1)$. Hence, we have: $\ln(1-\epsilon) \leq -\epsilon$. Replace this fact into the inequality $(*)$ we have:

$$\begin{aligned} & m \times (-\epsilon) \ \leq \ m \times \ln(1-\epsilon) \ \leq \ \ln\delta - \ln M(n,m) \\ \Leftrightarrow \ & m \times (-\epsilon) \ \leq \ \ln\delta - \ln M(n,m) \\ \Leftrightarrow \ & m \ \geq \ -\frac{\ln\delta}{\epsilon} + \frac{\ln M(n,m)}{\epsilon} \\ \Leftrightarrow \ & m \ \geq \ \frac{1}{\epsilon}\ln\frac{1}{\delta} + \frac{\ln M(n,m)}{\epsilon} \quad (**) \end{aligned}$$

$(**)$ shows that if we want to bound the worst case scenarios by some δ, we need a number of sample m greater or equal some value. For this value of m, the algorithm L will give us a ϵ-estimation hypothesis with probability $1-\delta$ over some sameple distribution D_*^m. Hence, L can be used as a PAC-learning algorithm for some concept class $\mathcal{C} \Rightarrow Q.E.D.$

Inconclusion, I have proved by construction that starting from assumptions made by **Theorem 1**, we can derive the requirement of m for algorithm L to be a PAC-learning algorithm.