PAC Learning under the Uniform Distribution

Purdue University
CS 590 SLA Lecture 9

Young-San Lin
February 16, Spring 2021

1 Preliminaries

We consider Probably Approximately Correct (PAC) learning under the uniform distribution. Suppose there is a concept class $\mathcal{C} := \{c : D \to R\}$ (or a set of functions), where $D = \{0,1\}^n$ and $R = \{0,1\}$. We can regard D as the space or domain of features or attributes and R as the space of labels. D can also be extended to $\mathbb{F}_p^n$ for some prime p.

A *target* $f \in \mathcal{C}$ is an underlying function that we would like to approximate. Given a *hypothesis* h which is not necessarily in $\mathcal{C}$[1], the *error* between h and f is

$$err_f(h) := \Pr_{x \leftarrow_u D}[f(x) \neq h(x)] \tag{1}$$

where $x \leftarrow_u D$ denotes that x is drawn uniformly at random (u.a.r.) from D. $err_f(h)$ denotes the probability that the outcome labels of f and h are different when x is picked u.a.r. from D. The error of h w.r.t. $\mathcal{C}$ is

$$err_{\mathcal{C}}(h) := \min_{f \in \mathcal{C}}\{err_f(h)\}. \tag{2}$$

This measures the minimum error between f and h among all f picked from $\mathcal{C}$, assuming that x is picked u.a.r. from D[2].

In a more general setting, an arbitrary distribution of D is $\mathcal{D}$ which might be unknown, and the error between f and h is

$$err_f(h) := \Pr_{x \leftarrow \mathcal{D}}[f(x) \neq h(x)]. \tag{3}$$

Here, we consider PAC learning under the uniform distribution throughout.

Definition 1.1. A uniform distribution learning algorithm on input ε and δ, is a randomized algorithm which has oracle access to $(x, f(x))$ where x is drawn uniformly and independently at random from D, such that it outputs h where $err_f(h) < \varepsilon$ with probability (w.p.) $1 - \delta$.

Intuitively, a learning algorithm samples $(x, f(x))$ multiple times without control over x, and the goal is to minimize the number of samples, so that w.p. $1 - \delta$, the error of approximating f is smaller than ε. For ease of notion, in this case, we say that the algorithm learns with accuracy $1 - \varepsilon$ and confidence $1 - \delta$.

Let m be the number of samples, our goal is to make $m = \mathsf{poly}(1/\varepsilon, 1/\delta, \log|D|)$. We are also interested in analyzing the running time. It is possible that even with small sample size, the learning algorithm takes exponential time. Quite commonly, the concept class $\mathcal{C}$ can be complicated and we might approximate $f \in \mathcal{C}$ by a succinct hypothesis h.

[1] If $h \in \mathcal{C}$, then this is *proper learning*. Otherwise, this is *improper learning*.

[2] I am not sure why we want to pick the closest function from $\mathcal{C}$ and compare it to the hypothesis h. There is some conceptual confusion here.

Theorem 1.2. Given ε, δ, and $\mathcal{C}$, there exists a randomized algorithm that takes $m = 1/\varepsilon(\ln|\mathcal{C}| + \ln(1/\delta))$ samples and learns with accuracy $1-\varepsilon$ and confidence $1-\delta$.

We note that this theorem does not have any assumption over the succincity of h and h can be in $\mathcal{C}$. The algorithm can take exponential time by an exhaustive search.

Proof. We propose the following algorithm:

1. Draw $m = 1/\varepsilon(\ln|\mathcal{C}| + \ln(1/\delta))$ samples, say $(x_1, f(x_1)), (x_2, f(x_2)), ..., (x_m, f(x_m))$.
2. Search through every $h \in \mathcal{C}$ and output the hypothesis h that agrees with each of the labels $f(x)$.

We want to compute the probability that $err_f(h) \geq \varepsilon$. We say that h is *bad* if $err_f(h) > \varepsilon$. Since h disagrees with f on ε fraction of D, for a fixed bad h, the probability that h survives the sampling test is

$$\begin{aligned}
\Pr[h(x_i) = f(x_i), \forall i \in [m]] &< (1-\varepsilon)^m && (4)\\
&= (1-\varepsilon)^{1/\varepsilon(\ln|\mathcal{C}|+\ln(1/\delta))} && (5)\\
&\leq \exp\left(-(\ln|\mathcal{C}| + \ln(1/\delta))\right) && (6)\\
&= \delta/|\mathcal{C}|. && (7)
\end{aligned}$$

There are at most $|\mathcal{C}|$ bad h's, so by union bound, the probability that at least one bad h survives the sampling test is δ. Thus, $err_f(h) < \varepsilon$, i.e., no h survives the test, w.p. $1-\delta$. □

Since $|\mathcal{C}|$ might be large, it is natural to ask if we can PAC-learn more efficiently when f belongs to a simpler concept class. For example, conjunctions, parities, tribes, dictators, and juntas.

2 Learning Conjunctions/Monomials

We start with the definition of conjunctions.

Definition 2.1. $c : \{0,1\}^n \to \{0,1\}$ is a *conjunction* if $c(x) = \bigwedge_{i\in S} x_i$ for some $S \subseteq [n]$. Namely, $c(x) = 1$ if and only if $x_i = 1$ for all $i \in [n]$.

The definition of monomials is equivalent to conjunctions.

Definition 2.2. $c : \{0,1\}^n \to \{0,1\}$ is a monomial if $c(x) = \Pi_{i\in S} x_i$ for some $S \subseteq [n]$. Namely, $c(x) = 1$ if and only if $x_i = 1$ for all $i \in [n]$.

By Theorem 1.2, we can PAC-learn conjunctions with $1/\varepsilon(\ln|\mathcal{C}| + \ln(1/\delta)) = 1/\varepsilon(n + \ln(1/\delta))$ samples since the size of the conjunction class is 2^n. The problem of interest is can we improve the term n?

Theorem 2.3. Given ε and δ, there exists a randomized algorithm that uses $O(1/\varepsilon\log(n/\delta) + 1/\varepsilon^2\log(1/\delta))$ samples and learns conjunctions on n variables with accuracy $1-\varepsilon$ and confidence $1-\delta$.

Proof. At a high level, we focus on the label 1 examples and retrieve the variables that belong to S. However, it is possible that S is small and it might suffice to return zero for all x. We have to estimate the fraction of label 1 instances in order to separate these cases.

We propose the following algorithm:

1. Draw $1/\varepsilon^2 \log(1/\delta)$ samples and estimate the fraction of label 1 examples, where the error differs by at most $\varepsilon/4$ w.p. $1 - \delta/2$. Let the estimate be f_1.

2. If $f_1 < \varepsilon/2$, then output $h(x) = 0$ for all x.

3. Otherwise, draw $m = O(1/\varepsilon \log(n/\delta))$ samples. Let

$$V = \{i \mid x_i = 1 \text{ for all sampled } x \text{ with } c(x) = 1\}.$$

4. Output $h(x) = \bigwedge_{i \in V} x_i$.

The first step estimates if S is small. If it is small, then step two outputs zero. If it is large, then we retrieve S by the third and the fourth step.

If $f(x) = 1$ for $\varepsilon/4$ fraction of x's, then $|f_1 - \varepsilon/4| < \varepsilon/4$ w.p. $1 - \delta/2$ by Chernoff's bound. The algorithm outputs zero in the second step.

Suppose $\Pr[f(x) = 1] > \varepsilon/4$. If $i \in S$, then $x_i = 1$ for all label 1 examples. If $i \notin S$, then the probability that $x_i = 1$ in each label 1 example is $1/2$ since it is independent of the label. We say that i is *bad* if it is not in S. Given that i is bad, the probability that $i \in V$ is

$$\begin{aligned}
\Pr[i \in V] &= \Pr[x_i = 1 \text{ in all label 1 examples}] && (8)\\
&= (1 - \Pr[x_i = 1 \mid i \notin S] \wedge \Pr[f(x) = 1])^m && (9)\\
&\leq \left(1 - \frac{1}{2} \cdot \frac{\varepsilon}{4}\right)^{8/\varepsilon \ln(n/\delta)} && (10)\\
&< \exp(-\ln(n/\delta)) = \delta/n. && (11)
\end{aligned}$$

By union bound over all n variable, we have that the probability that at least one bad i survives the sampling test is at most δ. □