1 PAC-Learning

In this lecture we will introduce the concept of PAC(Probably Approximately Correct)-Learning and give some basic properties of it. This concept was first defined and discussed by Valient in the 70’s and 80’s.

To define the concept, let $C$ be a class of boolean valued functions on the domain $\Omega$, that is $C \subseteq \{f : \Omega \rightarrow \{-1, 1\}\}$. This is the class we wish to learn.

Let $H$ be (another) class of boolean valued functions on $\Omega$. $H$ is called the hypothesis class, we will approximate (in the sense to be defined) functions in $C$ by functions in $H$.

In the following we will also have $\delta > 0$ and $\epsilon > 0$, in PAC, $\delta$ is related to Probably and $\epsilon$ is related to Approximately.

**Definition 1** We say that an algorithm $A$ PAC-learns the class $C$ using hypothesis $H$ if

$\forall \epsilon > 0, \delta > 0, f \in C, D \in \text{Prob}(\Omega)$

1. $A$ runs in time $\text{poly}(\text{sizeof}(f), 1/\delta, 1/\epsilon)$.

2. With prob. $\geq 1 - \delta$ over the samples $(X_i, f(X_i))$, where $\{X_i\}$ is chosen IID from $D$, $A$ outputs an $h \in H$ such that $D[f(x) \neq h(x)] < \epsilon$.

Where sizeof($f$) is a measure of the complexity of $f$, for example, the number of boolean operations in a small circuit which evaluates $f$.

In the sequel, when we discuss PAC-learning we will not always adhere to the polynomial time restriction but instead try to estimate the minimal time required to approximate a class $C$ (with some hypothesis class) in the above sense. When doing so, we shall often fix some $\epsilon$ or $\delta$ instead of working with all the possibilities for them.

2 PAC learning of the uniform distribution

In all that follows we will restrict ourselves to the following special context, we will take $\Omega = \{-1, 1\}^n$ and $D = U$ = the uniform distribution on $\{-1, 1\}^n$. 
Some special cases of PAC-learning are given names

1. Zero error learning : this is the case when $\epsilon = 0$ and $H = C$.
2. $\alpha(n)$ weak learning : $\epsilon = 1/2 - \alpha(n)$ and $\alpha(n) = 1/poly(n)$.
3. Membership query (MQ) : When the algorithm is allowed to choose the $\{X^i\}$ (there is no $D$ and $\delta$ is taken to be 0).

We will discuss two approaches to PAC-learning, information theoretical and running time. The information theoretical approach sometimes gives a lower bound for which it is very difficult to find an algorithm. The next example illustrates this approach.

**Example 2** $C = \{f : \{-1,1\}^n \rightarrow \{-1,1\}\}$.

**Claim 3** For $\epsilon = 0$ we need $2^n$ queries under the membership query model.

**Proof:** Clearly, the given function may differ from the function our algorithm outputs on the unevaluated inputs. $\square$

**Claim 4** For $\frac{1}{2} > \epsilon > 0$, we need at least $C(\epsilon)2^n$ queries under the membership query model.

**Proof:** Fix $X^1, X^2, \ldots, X^s$ the inputs seen. Then the algorithm can output at most $2^s$ functions. Let $B = \{\text{set of all possible output functions}\}$.

Define, for a function $f$, $B_\epsilon(f) = \{g \mid U[f \neq g] < \epsilon\}$. We can only learn the functions in $\bigcup_{f \in B} B_\epsilon(f)$, hence (since for any $f$ we have $|B_\epsilon(f)| = |B_\epsilon(0)|$) we need to have $2^s |B_\epsilon(0)| \geq 2^{2^n}$. But since

**Lemma 5** $|B_\epsilon(0)| \leq 2^{(1-C(\epsilon))2^n}$

The claim follows. $\square$

**Exercise 6** (1 Point) Prove the lemma, deduce that you cannot weakly learn all functions (for any $\alpha(n)$).

**Exercise 7** (1 Point) Show that the class of all monotone functions has size double exponential.
Proposition 8 The set of all monotone functions is $\frac{c}{n}$ weakly learnable for some constant $c$ (and some $H$).

Proof: Take $H = \{1, -1, x_1, x_2, \ldots, x_n\}$. Divide into cases

1. Easy case, when $f$ is not balanced. Take $\frac{1000n^2}{\delta^2}$ queries, if $\hat{E}f \in \left[\frac{-1}{16}, \frac{1}{16}\right]$ then output $\text{sgn}(\hat{E}f)$. By large deviations, if $Ef \in \left[\frac{-1}{16}, \frac{1}{16}\right]$ then w.p. $\geq 1 - \delta$ $\hat{E}f \in \left[\frac{-1}{8}, \frac{1}{8}\right]$ and $\text{sgn}(Ef) = \text{sgn}(\hat{E}f)$. This shows that when $f$ is so unbalanced that $Ef \not\in \left[\frac{-1}{16}, \frac{1}{16}\right]$ then the above algorithm performs as required, on the other hand, if $f$ is relatively balanced so that $Ef \in \left[\frac{-1}{16}, \frac{1}{16}\right]$ then the above case will w.p. $\geq 1 - \delta$ not be picked by the algorithm. In the remaining case $Ef \in \left(\left(\left[\frac{-1}{4}, -\frac{1}{16}\right]\cup \left[\frac{1}{16}, \frac{1}{4}\right]\right)\right)$ it doesn’t matter if the above case is executed by the algorithm, or the next case.

2. When $Ef \in \left[\frac{-1}{4}, \frac{1}{4}\right]$ we get by Harper’s inequality that $\sum I_i(f) \geq 1$ (we actually get something even better from the inequality). Hence there exists $i$ with $I_i(f) \geq \frac{1}{n}$. Since for monotone functions the $\{i\}$'th Fourier coefficient equals the $i$'th influence it follows that $P(f(x) \neq x_i) \leq \frac{1}{2} - \frac{1}{n}$, so our algorithm would perform well if it outputs $h = x_i$ in this case. We will actually not necessarily output this $x_i$, but the $x_j$ we do output will also work well for us as detailed in the following.

Take $\frac{n^3}{\delta}$ samples and estimate for each $j$, $\hat{P}(f(x) = x_j)$. If found $j$ s.t. $\hat{P}(f(x) = x_j) \geq \frac{1}{2} + \frac{1}{2n}$ then output $h = x_j$. By another large deviation calculation, w.p. $\geq 1 - \delta$ we will find a $j$ for which $P(f(x) = x_j) \geq \frac{1}{2} + \frac{1}{4n}$ and hence the algorithm works with constant $c = \frac{1}{4}$ in this case.

$\square$

Proposition 9 The set of all monotone functions can be learned in time $\frac{1}{\delta}2^{O(\sqrt{n}\log(n)/\epsilon)}$.

The proof will be given in the next lecture.