0368.4163 Randomness and Computation June 10, 2009

Lecture 12

Lecturer: Ronitt Rubinfeld *Scribe: efrat bank, Guy Harari, Yehudit Hasson*

1 Introduction

In this lecture we prove that a "weak" PAC learning algorithm implies a "strong" PAC learning algorithm. Recall the definition that was given in previous lectures for PAC:

Definition 1 *An algorithm $\mathcal{A}$ (**"strongly"**) PAC learns a concept class $\mathcal{C}$ if $\forall c \in \mathcal{C}, \forall$ distribution $\mathcal{D}, \forall \varepsilon > 0$, with probability greater or equal $\frac{3}{4}$, given examples of c which are chosen according to the distribution $\mathcal{D}$,* A *outputs h such that*

$$\Pr_{\mathcal{D}}[h(x) \neq c(x)] \leq \varepsilon. \tag{1}$$

Remarks

- Note that in previous lecture the parameter δ was introduced, and the requirement was that with probability greater or equal $1-\delta$ $\mathcal{A}$ output a "good" hypothesis. In homework we have shown that it is enough to consider the case of $\delta = 3/4$, and results for general delta can be achieved via amplification.
- In previous lectures the distribution $\mathcal{D}$ was chosen to be the uniform distribution. In fact, any distribution may be used.

Definition 2 *An algorithm $\mathcal{A}$ **weakly** PAC learns a concept class $\mathcal{C}$ if $\forall c \in \mathcal{C}$, $\exists \gamma > 0$ such that $\forall$ distribution $\mathcal{D}$, with probability $\geq \frac{3}{4}$, given examples of c which are chosen according to the distribution $\mathcal{D}$, $\mathcal{A}$ outputs h such that*

$$\Pr_{\mathcal{D}}[h(x) \neq c(x)] \leq \frac{1}{2} - \frac{\gamma}{2}. \tag{2}$$

- The term $\frac{\gamma}{2}$ is called the ***advantage*** of $\mathcal{A}$.
- We will consider functions $c \in$ C such that $c : \{\pm 1\}^n \longrightarrow \{\pm 1\}$

The main result prove in this lecture is:

Theorem 1 *If a concept class $\mathcal{C}$ can be weakly learned, then $\mathcal{C}$ can be strongly learned.*

2 The Algorithm

2.1 The Intuition

The main idea is to run the weak-learning algorithm first with the uniform distribution (though it does not matter what is the initial distribution) in order to get an hypothesis h_1. Then, we would like **not** to choose the "good" examples again, meaning to give them a small probability, and to run the algorithm again with the new examples to get another hypothesis h_2. Next, we would like to somehow combine between the two output functions. A natural way would be to give the examples that were "good" for both h_1 and h_2 a low probability, and give the "bad" examples a higher one. We will refer to this process as "filtering". The following figure illustrates the main idea of the algorithm.

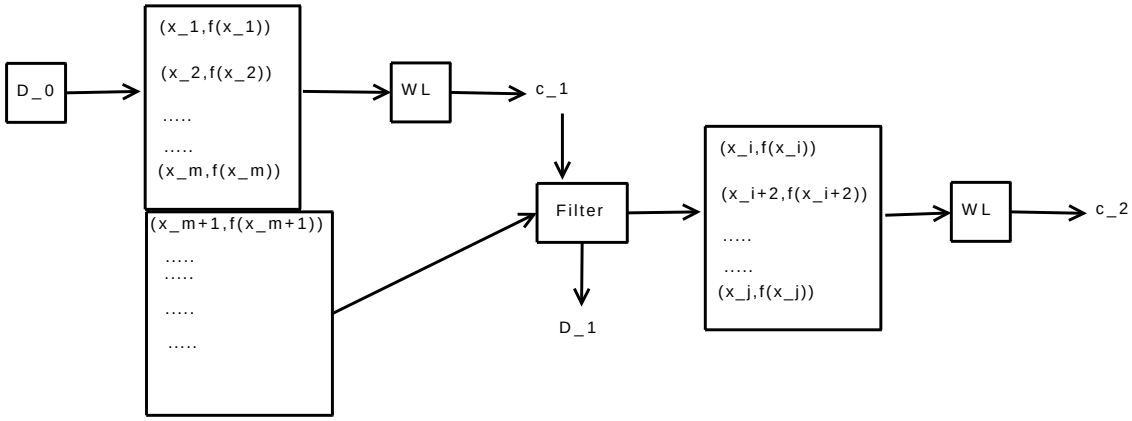

Figure 1: Flow of the algorithm

2.2 The Algorithm

Given a weak learner WL, a distribution $\mathcal{D}$, a concept $f : \{\pm 1\}^n \longrightarrow \{\pm 1\}$ and parameters ε and γ, the algorithm goes as follows: (We illustrate the case for the uniform distribution. Note that the algorithm can be easily modified to any initial distribution although we are not showing it here.)

Stage 0 - Initialization:

Set distribution $\mathcal{D}_0 \longleftarrow \mathcal{D}(=\mathcal{U})$

Use WL to generate (with high probability) an hypothesis c_1 such that $\Pr_{\mathcal{D}_0}[f(x) = c_1(x)] \geq \frac{1}{2} + \frac{\gamma}{2}$

Stage i:

(1) Construct $\mathcal{D}_i$ via the **filtering mechanism**

(2) Run WL with examples from $\mathcal{D}_i$ to get hypothesis c_{i+1} such that $\Pr_{\mathcal{D}_i}[f(x) = c_{i+1}(x)] \geq \frac{1}{2} + \frac{\gamma}{2}$

(3) If $\Pr_{\mathcal{D}_i}[f(x) = Maj(c_1, ..., c_{i+1})] \geq 1 - \varepsilon$ output $\text{Maj}(c_1, ..., c_{i+1})$

After $\mathsf{T} = \mathsf{O}(\frac{1}{\gamma^2\varepsilon^2})$ **steps:**

output c=$\text{Maj}(c_1, ..., c_T)$

Filtering procedure:

Given $(x, f(x))$ chosen according to the distribution D_0:

If $\text{Maj}(c_1, \ldots, c_i)$ is wrong on x, keep x

If # of c_i's right - # of c_i's wrong $\geq \frac{1}{\varepsilon\gamma}$, throw x away

Else # of c_i's right - # of c_i's wrong $= \frac{\alpha}{\varepsilon\gamma}$, for some $0 < \alpha < 1$, keep x with probability $1 - \alpha$

Remarks

- We need to add the condition that if the filtering process takes too long, i.e. $O(\frac{1}{\varepsilon})$, stop early and output $\text{Maj}(c_1, ..., c_j)$. This is correct since if one cannot find examples, i.e. all (or most) examples should be tossed, this means that the error must be small and the algorithm may stop.
- Until i is about $\gamma\varepsilon$ none of the examples is being tossed.
- If x is tossed at some stage, it can still appear with some probability in one of the following stages.

3 Preliminaries

Here are some notations and their properties:

1. $R_c(x) = \begin{cases} +1 & \text{if } f(x) = c(x) \\ -1 & \text{o.w.} \end{cases}$ gives +1 if (weak) hypothesis c is right on example x
2. $N_i(x) = \sum_{1 \le j \le i} R_{c_j}(x)$ is the number of right c's exceeding the wrong ones (i.e. #right-#wrong)
3. $M_i(x) = \begin{cases} 1 & \text{if } N_i(x) \le 0 \\ 0 & \text{if } N_i(x) \ge \frac{1}{\varepsilon\gamma} \\ 1 - \varepsilon\gamma N_i(x) & \text{o.w.} \end{cases}$
   is a "Probabilty filter keeps sample x"- a "measure" which upper bounds the error of hypothesis $c = \text{Maj}(c_1, \ldots, c_i)$ on example x.
4. $|M_i| = \sum_x M_i(x)$ is the total "mass" of all examples according to "measure" M_i.
5. $D_{M_i}(x) = \frac{M_i(x)}{|M_i|}$ is a distribution over x given M_i.
6. Observe that $error(c_{i+1}) \equiv Pr_{x \in uniform}[c(x) \neq f(x)] \le \frac{|M_i|}{2^n}$
   the error of concept class c is the number of inputs for which the majority gaves us a wrong answer and it's smaller than $|M_i|$. (we noramlized it by the total number of inputs).
7. $\text{Adv}_c(M) = \sum_x R_c(x)M(x)$ is the advantage of c on M which gives an indication on the number of inputs for which c is correct. (Random guessing gives 0.)
8. $\text{Adv}_c(M) \ge \gamma|M|$ iff $\Pr_{x \in D_M}[c(x) = f(x)] \ge \frac{1}{2} + \frac{\gamma}{2}$
9. Note that if $\Pr_{x \in D_M}[c(x) = f(x)] \ge \frac{1}{2} + \frac{\gamma}{2}$ and $|M| \ge \varepsilon 2^n$, then $\text{Adv}_c(M) \ge \varepsilon\gamma 2^n$

4 Proof of correctness

Let's define $A_i(x) = \sum_{0 \le j \le i-1} R_{c_{j+1}}(x) M_j(x)$

Claim 2 $A_i(x) \le \frac{1}{\varepsilon\gamma} + \frac{\varepsilon\gamma}{2} \cdot i$

Before proving this claim, we first use it to bound the maximum number of iterations required by the boosting algorithm. Hence, if a concept can be weakly PAC learned, then it can be ("strongly") PAC learned.

Claim 3 *The maximum number of iterations required by the boosting algorithm is* $i_0 \le \frac{2}{\gamma^2\varepsilon^2}$.

Proof We prove the claim by showing that assuming the algorithm does not stop after $i_0 + 1$ stages, then $error(c_{i_0}) \ge \varepsilon$ i.e. $\forall j |M_i| \ge \varepsilon 2^n$ and this leads to a contradiction.

Suppose the claim is not true and $i_0 > \frac{2}{\gamma^2\varepsilon^2}$ then we will consider the 2^n by $(i_0 + 1)$ dimensional matrix, where the rows are labelled by inputs and the columns by iteration number, and in which the entry corresponding to x,i contains $A_i(x)$. We first bound the total sum of the entries of this matrix from below by showing that the total sum of each column is large. The latter follows from Observation 8 above, which follows from the correctness of the weak learning algorithm and from the fact that the measure on which the algorithm errs is still large. On the other hand, using the claim we can bound from above the total sum of the matrix by upper bounding the total sum of any row. The claim, which we prove later, shows that after a number of iterations, there is a nontrivial upper bound on the number of hypotheses (weighted by the corresponding measures) which err on this input. Note that if the majority

of the hypotheses do well on an input, then its measure will go to zero, but later on, if too many of the newer hypotheses start to do badly on the same input, then its measure will become nonzero again.

$$\begin{aligned}
\sum_x A_{i_0+1}(x) &= \sum_x \sum_{0\le j\le i_0} R_{c_{j+1}}(x)M_j(x) && (3)\\
&= \sum_{0\le j\le i_0} \underbrace{\sum_x R_{c_{j+1}}(x)M_j(x)}_{Adv_{c_{j+1}}(M_j(x))} && (4)\\
&\ge (i_0+1)\varepsilon\gamma 2^n && (5)
\end{aligned}$$

Using Claim 2 leads to an upper bound:

$$\begin{aligned}
\sum_x A_{i_0+1}(x) &\le \sum_x (\frac{1}{\varepsilon\gamma} + \frac{\varepsilon\gamma}{2}(i_0+1)) && (6)\\
&= 2^n(\frac{1}{\varepsilon\gamma} + \frac{\varepsilon\gamma}{2}(i_0+1)) && (7)
\end{aligned}$$

Using both bounds, $(i_0+1)\gamma 2^n\varepsilon \le \sum_x A_{i_0+1}(x) < 2^n(\frac{1}{\varepsilon\gamma} + \frac{\varepsilon\gamma}{2}(i_0+1)) \Rightarrow i_0 \le \frac{2}{\gamma^2\varepsilon^2} - 1$, we arrive at a contradiction. So, the algorithm must run for $\frac{2}{\gamma^2\varepsilon^2}$ iterations or less. ■

Fact 4 (The Elevator Principle) *If one rides an elevator from the ground floor, then one ascends from the k-th to the $(k+1)$-th floor at most 1 more time than one descends from the $(k+1)$-th to the k-th floor. (Analogous argument holds when traveling from the ground floor to basements.)*

Proof of Claim 2: The process of adding each term of $N_i(x)$ corresponds to an elevator ride with $R_{c_j}(x)$ dictating the direction and partial sum $N_j(x)$ denoting the current level. The plan is to first match pairs of $R_{cj+1}(x)M_j(x)$ terms and obtain an upper bound of their sum using properties of function $M_j(x)$. As for the unmatched pairs, we can bound the number of them (using the Elevator Argument) and also their sums. And so, an upper bound for $A_i(x)$ can be obtained.

Matched Pairs

For each $k \ge 0$,
match j such that $N_j(x) = k$ and $N_{j+1}(x) = k+1$
with j' such that $N_{j'}(x) = k+1$ and $N_{j'+1}(x) = k$
(analogously match -k to -(k+1) with -(k+1) to -k)

For each matched pair of terms corresponding to indices $a = j, b = j'$, the sum is
$\underbrace{R_{c_{a+1}}(x)}_{+1}\underbrace{M_a(x)}_{N_a(x)=k} + \underbrace{R_{c_{b+1}}(x)}_{-1}\underbrace{M_b(x)}_{N_b(x)=k+1} = M_a(x) - M_b(x)$.

If $0 \le k \le \frac{1}{\varepsilon\gamma}$ or $0 \le k+1 \le \frac{1}{\varepsilon\gamma}$, then
$M_a(x) - M_b(x) \le \varepsilon\gamma$ (because $\frac{M_b(x)-M_a(x)}{k+1-k}$ is the slope of $M_i(x)$ which is $\ge -\varepsilon\gamma$),
else
$(k < -1$ and $M_a(x) = M_b(x) = 0)$ or $(k > \frac{1}{\varepsilon\gamma}$ and $M_a(x) = M_b(x) = 1) \Rightarrow M_a(x) - M_b(x) = 0$

Therefore, the total contribution of matched pairs is $\le 0.5\varepsilon\gamma i$ (because $A_i(x)$ has i pairs).

Unmatched Terms Notice that unmatched pairs are in the "same direction", i.e. all $R_{c_j}(x)$'s are either negative or positive.

Suppose all $R_{c_j}(x)$'s are negative (i.e. -1), then their contribution to the sum is negative (because each term becomes $-M_j(x) \leq 0$). So they do not loosen the upper bound we already derived from matched pairs.

Suppose all $R_{c_j}(x)$'s are positive (i.e. $+1$). Then $N_j(x) \geq 0$, and so each term is $M_j(x) = 1 - \varepsilon\gamma N_j(x)$ if $N_j(x) \in [0, \frac{1}{\varepsilon\gamma}]$ and 0 otherwise. The Elevator Lemma tells us that there is at most one unmatched $N_j(x)$ for each integer value in the interval $[0, \frac{1}{\varepsilon\gamma}]$, and so the total contribution of them (sum of a arithmetic series from 0 to 1 with $\frac{1}{\varepsilon\gamma}$ terms) is $\leq \frac{1}{2\varepsilon\gamma} < \frac{1}{\varepsilon\gamma}$

Summing up the total contribution from both matched and unmatched terms gives

$$A_i(x) < \underbrace{\frac{1}{\varepsilon\gamma}}_{unmatched} + \underbrace{\frac{\varepsilon\gamma i}{2}}_{matched}.$$

■