6.842 Randomness and Computation April 25, 2012

Lecture 20

Lecturer: Ronitt Rubinfeld *Scribe: Jon Schneider*

1 Introduction

In this lecture, we will demonstrate a procedure to convert any weak learning algorithm to a strong learning algorithm. This procedure is known as the Boosting Algorithm.

Before we state the Boosting Algorithm, let us recall the definitions of weak and strong learning algorithms.

Definition 1 *The algorithm A_W is a **weak PAC learning algorithm** for the concept class $\mathcal{C}$ if for every concept $f \in \mathcal{C}$ and distribution $\mathcal{D}$ over the domain of f, there exists some $\gamma > 0$ for every $\delta > 0$, such that the algorithm with access to examples $(x, f(x))$ with x drawn from $\mathcal{D}$, returns (with probability at least $1 - \delta$) a concept c such that*

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

Definition 2 *The algorithm A_S is a **strong PAC learning algorithm** for the concept class $\mathcal{C}$ if for every concept $f \in \mathcal{C}$, input distribution $\mathcal{D}$ and $\delta, \epsilon > 0$, the algorithm with access to examples $(x, f(x))$ drawn from $\mathcal{D}$, returns (with probability at least $1 - \delta$) a concept c such that*

$$\Pr_{\mathcal{D}}[f(x) \neq c(x)] \leq \epsilon$$

Our main theorem is the following.

Theorem 3 *If a concept class $\mathcal{C}$ is weakly learnable, then it is strongly learnable.*

2 Boosting Algorithm

To prove Theorem 3, we will demonstrate a procedure (called the *Boosting Algorithm*) that transforms a weak learning algorithm into a strong learning algorithm. Below, we will let f be our unknown concept, $\mathcal{D}$ our initial distribution (which we assume to be uniform)[1] , and WL be our weak learning algorithm.

The algorithm runs in various stages, which we describe below.

- **Initial Stage**: Set $\mathcal{D}_0 = \mathcal{D}$ and use our weak learning algorithm WL to generate a concept c_0 such that $\Pr_{\mathcal{D}_0}[f(x) = c_1(x)] \geq \frac{1}{2} + \frac{\gamma}{2}$.
- **Stage** i: Given concepts $c_1, c_2, \ldots, c_i$ from the previous i stages, we construct the distribution $\mathcal{D}_i$ via our **filtering procedure**. We then run WL on $\mathcal{D}_i$ to output a concept c_{i+1} such that $\Pr_{\mathcal{D}_i}[f(x) = c_{i+1}(x)] \geq \frac{1}{2} + \frac{\gamma}{2}$.
- **Final Stage**: After $T = O(1/\gamma^2\epsilon^2)$ stages, or whenever $\text{maj}(c_1, c_2, \ldots, c_i)$ is correct on a fraction of at least $(1 - \epsilon)$ of the inputs, output the concept $c = \text{maj}(c_1, c_2, \ldots, c_i)$.

We used a filtering procedure to construct the distribution $\mathcal{D}_i$. We now describe the algorithm for the following filtering procedure, which we repeat until we have the desired number of samples.

- Draw an example $(x, f(x))$ from the uniform distribution $\mathcal{D}$.

[1] It turns out that all of our results also hold for the general case where the input distribution is arbitrary (as opposed to uniform). For sake of convenience, however, in this lecture we will assume the input distribution is uniform over the domain of f.

- If $\text{maj}(c_1, c_2, \ldots, c_i)$ disagree with $f(x)$, keep this sample.
- Let n_r equal the number of j such that $c_j(x) = f(x)$, and let n_w equal the number of j such that $c_j(x) \neq f(x)$ (so $n_r + n_w = i$). If $n_r - n_w \geq \frac{1}{\gamma\epsilon}$, then toss the sample.
- Otherwise, if $n_r - n_w = \frac{\alpha}{\gamma\epsilon}$ for some $0 \leq \alpha < 1$, toss the sample with probability α (and keep it otherwise).

If we look at this filtering procedure in terms of the number n_w of incorrect concepts, then we can see that the probability p of keeping a sample from $\mathcal{D}$ depends piecewise-linearly on the value of n_w; specifically, between 0 and $\frac{i}{2} - \frac{1}{2\gamma\epsilon}$ this function equals 0, from $\frac{i}{2} - \frac{1}{2\gamma\epsilon}$ to $\frac{i}{2}$ it increases with slope $2\gamma\epsilon$ from 0 up to 1, and from $\frac{i}{2}$ to i it stays at 1.

The main claim we will want to show is that if we need at least $\frac{1}{\epsilon}$ samples of $\mathcal{D}$ to output a sample of $\mathcal{D}_i$, then $\text{Maj}(c_1, c_2, \ldots, c_i)$ is $(1-\epsilon)$-close to f. We will do this in the following two sections.

3 Notation

In the following section, we will establish (most of) a proof of correctness for the Boosting Algorithm. First, however, we will introduce some terminology that will be useful in our following proof.

Definition 4 *For a concept c and an input x,*

$$R_c(x) = \begin{cases} 1, & \text{if } f(x) = c(x) \\ -1, & \text{if } f(x) \neq c(x) \end{cases}$$

Definition 5 *If c_i is the concept introduced at stage i of the Boosting Algorithm, we define*

$$N_i(x) = \sum_{1 \leq j \leq i} R_{c_i}(x)$$

Note that $N_i(x)$ is equal to the value $n_r - n_w$ we saw earlier in the filtering algorithm.

Definition 6 *We define*

$$M_i(x) = \begin{cases} 1, & \text{if } N_i(x) \leq 0 \\ 0, & \text{if } N_i(x) \geq \frac{1}{\epsilon\gamma} \\ 1 - \epsilon\gamma N_i(x), & \text{otherwise} \end{cases}$$

Similarly, note that $M_i(x)$ is nothing more than the probability of the filtering procedure at stage i keeping x.

Definition 7 *Define*

$$|M_i(x)| = \sum_x M_i(x)$$

Definition 8 *Define*

$$D_{M_i}(x) = \frac{M_i(x)}{|M_i|}$$

The definition for $D_{M_i}(x)$ captures the actual probability of x in the distribution $\mathcal{D}_i$ returned by this filtering procedure. We can think of $|M_i|$ as the total "mass" of all of the $M_i(x)$ probabilities.

Definition 9 *We define the advantage of a concept c over one of the distributions M_j as*

$$\mathrm{Adv}_c(M_i) = \sum_x R_c(x) M_j(x)$$

Note that we can construct a large two-dimensional 2^n by i matrix, where the rows are indexed by possible inputs x, where the columns are indexed by the number j of the current stage, and where element (x, j) is given by $R_{c_{j+1}}(x)M_j(x)$. If this is the case, note that $\mathrm{Adv}_{c_{j+1}}(M_i)$ is just the sum of the jth column of this matrix. It is also interesting to consider the row sums of this matrix.

Definition 10 *Let*

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

These $A_i(x)$ are the row sums of the xth row of the above matrix.

We will also need a simple fact about the advantage.

Theorem 11 *If $\Pr_{x \in \mathcal{D}_i}[c(x) = f(x)] \ge \frac{1}{2} + \frac{\gamma}{2}$, then $\mathrm{Adv}_c(M_i) \ge \gamma |M|$.*

Proof Note that $\mathrm{Adv}_c(M_i) = |M_i|(\Pr[c_i \text{ is right}] - \Pr[c_i \text{ is wrong}]) = |M_i|(2\Pr[c_i \text{ is right}] - 1) \ge \gamma |M_i|$. ■

Corollary 12 *If $\Pr_{x \in \mathcal{D}_i}[c(x) = f(x)] \ge \frac{1}{2} + \frac{\gamma}{2}$ and $|M| \ge \epsilon 2^n$, then $\mathrm{Adv}_c(M_i) \ge \gamma \epsilon 2^n$.*

4 Correctness of the Boosting Algorithm

In this section, we will provide most of a proof of correctness for the Boosting Algorithm. Our proof will rely on the following theorem, which we will prove next lecture.

Theorem 13 *For all x we have that*

$$A_i(x) \le \frac{1}{\epsilon\gamma} + \frac{\epsilon}{\gamma} 2i$$

Proof [Proof of Theorem 3]

Assume our algorithm has not terminated after i steps. If this is true, then $|M_i| \ge \epsilon 2^i$, or otherwise the majority of the c_i would agree with f at least $(1 - \epsilon)$ of the time and our algorithm would have already terminated.

Now, by Theorem 11, we have that

$$\begin{aligned} \sum_x A_{i+1}(x) &= \sum_x \sum_{0 \le j \le i} R_{c_{j+1}}(x) M_j(x) \\ &= \sum_{0 \le j \le i} \mathrm{Adv}_{c_{j+1}}(M_j) \\ &\ge (i+1)\epsilon\gamma 2^n \end{aligned}$$

On the other hand, by our unproven Theorem 13 above, we have that

$$\sum_x A_{i+1}(x) \le 2^n \left(\frac{1}{\epsilon\gamma} + \frac{\epsilon\gamma}{2}(i+1) \right)$$

Combining these two bounds, we get that

$$(i+1)\epsilon\gamma 2^n \leq 2^n \left(\frac{1}{\epsilon\gamma} + \frac{\epsilon\gamma}{2}(i+1) \right)$$

Simplifying and solving for i, we find that

$$i \leq \frac{2}{\epsilon^2\gamma^2} - 1 = O\left(\frac{1}{\epsilon^2\gamma^2} \right)$$

This shows that after $\frac{2}{\epsilon^2\gamma^2}$ stages, our algorithm should terminate. ■

5 Next Lecture

Next lecture, we will prove Theorem 13 above. The main idea behind the proof will be "Elevator Lemma", which says that if you take an elevator ride, the number of times you go up from the ith floor to the $i+1$th floor and the number of times you go down from the $i+1$th floor to the ith floor differ by at most 1. We will use an analogous cancellation involving the N_i functions to establish the lower bound in Theorem 13.

We will also discuss the connection between the Boosting Algorithm, strong and weak PAC learning, and Yao's lemma for hardcore predicates.