6.842 Randomness and Computation April 9th and 14th, 2014

Lecture 17 and 18

Lecturer: Ronitt Rubinfeld *Scribe: Zeyuan Allen-Zhu*

The goal of these two lectures are to study boosting, a general technique that turns a distribution-free weak learner into a strong learner.

Definition 1 *An algorithm A* weakly PAC learns *a concept class C (with confidence $\delta > 0$ and error $\gamma > 0$) if for all $f \in C$ and for* all distribution $\mathcal{D}$*, with probability at least $1 - \delta$ and given samples from f, A outputs some c satisfying*

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

We emphasize here that the weak learning algorithm A does not know the distribution $\mathcal{D}$. It only sees samples from $\mathcal{D}$. The time and sampling complexity of A usually depends on δ and γ. The definition of weak learner is different from the normal but strong version of PAC learning in terms of the error guarantee: the strong version requires that for any $\varepsilon > 0$, the error $\Pr_{\mathcal{D}}[f(x) \neq c(x)]$ can be made as small as ε.

(Recall that we have learned from the previous lecture on how to obtain a weak leaner for monotone functions, but only when the distribution is uniform. If that weak leaner worked for all distributions (i.e., is distribution-free), it would imply that monotone functions can be PAC learned in the strong sense, contradicting to the impossibility result.)

1 Thought Experiments

We begin with some intuitive trials on how to turn a (distribution-free) weak leaner into a strong one.

The first (and very bad) trial is to run the weak leaner multiple times on the same distribution $\mathcal{D}$, and output for instance the majority of the predictions. This approach fails very miserably because typically in learning, repetition can only improve the confidence δ (see for instance Homework 7-3), but not the prediction accuracy. A simple example to illustrate this phenomenon is to consider some weak learner that always outputs the same prediction c no matter how many times it is run. In such a case, the majority of the same function c is still c itself, and therefore the prediction accuracy is not improved at all.

The second (and very promising) trial is to always run a weak learner on the samples where the previous learners fail to predict. More specifically, suppose that from distribution $\mathcal{D}$ the weak learner A obtains some prediction $c_1(x)$ (from samples $(x_1, f(x_1)), (x_2, f(x_2)), \dots (x_m, f(x_m))$ if m samples are needed in A). Next, we query $\mathcal{D}$ for more examples $(x_{m+1}, f(x_{m+1})), \dots$, but only focus on those x_i such that $c_1(x_i) \neq f(x_i)$. In other words, we use $c_1(x)$ to *filter out* the successfully predicted samples, and only focus on the rest. This gives rise to a different distribution $\mathcal{D}_1$, and we can pass it to the weak learner A to produce a second prediction $c_2(x)$. This process can go on for a number of stages.

The second trial looks very promising, because it uses the distribution-free feature of the weak leaner A, and always refines on the samples that are mistakenly predicted. However, a fundamental problem exists: how do we, in the end of the algorithm, combine the sequence of predictions $c_1(x), c_2(x), \dots$? For instance, given some new sample x and suppose we want to predict the unknown label $f(x)$, how do we choose from the answers of $c_1(x), c_2(x), \dots$? The filtering techniques allows us to choose the $c_j(x)$ among all possible j only after seeing the ground-truth label $f(x)$, so for new and to-be-predicted samples, the filtering no longer helps.

In this lecture, we are going to study the *boosting* algorithm that is essentially built from the second trial above. In fact, it chooses the majority of $c_1(x), c_2(x), \dots$ as the final output (of the strong leaner), but changes the definition of filtering to ensure the correctness.

(One may ask if there are indeed example of concept classes where it is easy to design weaker learners but hard to design for strong ones. In fact, it is not clear if such examples exist so the final theorem

of the boosting algorithm turns out to be a theoretical result. However, the strong learner of DNF was indeed introduced using boosting by Jackson: he uses weak learners that are not distribution-free, but work for a sufficiently large class of distributions that is sufficient for the boosting of DNF. Also, variants of boosting have found numerous applications in practical problems such as character recognitions.)

2 The Boosting Framework

Pick δ to be smaller than $1/T$ where T is the number of the phases, and suppose we are given samples from $\mathcal{D}$ that are labelled according to f.

- Stage 0: initialization. $\mathcal{D}_0 \leftarrow \mathcal{D}$. Run the weak learner on $\mathcal{D}_0$ to generate $c_1(x)$ satisfying $\Pr_{\mathcal{D}_0}[f(x) = c_1(x)] \geq \frac{1}{2} + \frac{\gamma}{2}$.
- For stage $i \leftarrow 1$ to $T = O(1/\gamma^2\varepsilon^2)$.
  - Stop the algorithm if $Maj(c_1, c_2, \ldots, c_i)$ is correct on $1-\varepsilon$ fraction of the inputs with respect to $\mathcal{D}$, and output the function $Maj(x_1, \ldots, c_i)$ as the final prediction.

    (This step can be done by taking $O(1/\varepsilon)$ samples from $\mathcal{D}$ and checking how many of them fail the test on $f(x) = Maj(c_1(x), \ldots, c_i(x))$.)
  - Construct $\mathcal{D}_i$ via some "filtering procedure".

    (This will be introduced in the next section, but from a high level, $\mathcal{D}_i$ is constructed from $\mathcal{D}$ by favoring samples where the previous predictions are incorrect. It will be constructed in a probabilistic way based on how many previous predictions are correct.)
  - Run the weak leaner with distribution $\mathcal{D}_i$ and get a new function $c_{i+1}(x)$ satisfying $\Pr_{\mathcal{D}_i}[f(x) = c_{i+1}(x)] \geq \frac{1}{2} + \frac{\gamma}{2}$.
- If the algorithm does not stop in T stages, output $C = Maj(c_1, \ldots, c_T)$.

  (Our theorem ensures that the algorithm will never reach here, at least with reasonable probability.)

3 The Specific Choice of Filtering Procedure

At stage i of the algorithm, given predictions $c_1, \ldots, c_i$ from stage 0 through stage $i-1$, we construct $\mathcal{D}_i$ as follows. Given a sample $(x, f(x))$ from $\mathcal{D}$, we compare $Maj(c_1(x), \ldots, c_i(x))$ and $f(x)$.

- If $Maj(c_1(x), \ldots, c_i(x)) \neq f(x)$, we keep this sample.
- Otherwise, letting $\#right$ be the number of correct predictions among $c_1(x), \ldots, c_i(x)$, and $\#wrong$ be the number of incorrect predictions (so we have $\#right + \#wrong = i$).

  If a large majority is correct —that is, $\#right - \#wrong > \frac{1}{\gamma\varepsilon}$ (which is equivalent to $\#wrong \leq \frac{i}{2} - \frac{1}{2\gamma\varepsilon}$)— we discard this sample.
- If $\#right - \#wrong = \frac{\alpha}{\gamma\varepsilon} < \frac{1}{\gamma\varepsilon}$ for some $\alpha \in [0, 1]$, we keep this sample with probability $1 - \alpha$.

This above random procedure (of generating samples) gives the definition of $\mathcal{D}_i$. Note that when i is small, the threshold $\frac{i}{2} - \frac{1}{2\gamma\varepsilon}$ is negative so nearly all samples from $\mathcal{D}$ are kept, and in other words, $\mathcal{D}_i \approx \mathcal{D}$ for small i.

As another remark, in principle, we need to make sure that the weak learner receives enough samples in each stage i; or in other words, the sampling complexity does not blow up from weak to strong learner. This is true because, at stage i, the non-stopping criterion $\Pr_{\mathcal{D}}[Maj(c_1(x), \ldots, c_i(x)) = f(x)] < 1 - \varepsilon$ ensures that we only need $\leq \frac{1}{\varepsilon}$ samples from $\mathcal{D}$ in order to generate a sample from $\mathcal{D}_i$.

4 Sketched Proof of the Correctness

We mostly only introduce some notations here, and the full proof will be given in the next lecture. Let $R_c(x)$ capture the correctness of a prediction c at input x:

$$R_c(x) := \begin{cases} +1, & \text{if } f(x) = c(x); \\ -1, & \text{otherwise.} \end{cases}$$

Let $N_i(x)$ indicate "$\#correct - \#wrong$" for the first $i \geq 0$ predictions

$$N_i(x) := \sum_{1 \leq j \leq i} R_{c_j}(x) \ ,$$

and let $N_i(x) = 0$ for $i = 0$. For any $i \geq 0$, we also denote by

$$M_i(x) := \begin{cases} +1, & \text{if } N_i(x) \leq 0; \\ 0, & \text{if } N_i(x) \geq \frac{1}{\gamma\varepsilon}; \\ 1 - \varepsilon\gamma \cdot N_i(x), & \text{otherwise.} \end{cases}$$

so that the distribution $\mathcal{D}_{M_i}(x) := \frac{M_i(x)}{\sum_x M_i(x)}$ coincides with our distribution $\mathcal{D}_i$. We emphasize again that this distribution $\mathcal{D}$ includes all incorrectly predicted x plus some others.

We define also the advantage of a prediction $c(x)$ over M_i as

$$Adv_c(M_i) := \sum_x R_c(x) \cdot M_i(x) \ .$$

It is clear from the definition that $\Pr_{x \in \mathcal{D}_{M_i}}[c(x) = f(x)] = \frac{1}{2} + \frac{Adv_c(M_i)}{2\sum_x M_i(x)}$. Note that, by the definition of the weak learner, we have that the special choice of $c = c_{i+1}$ makes sure that $\Pr_{x \in \mathcal{D}_{M_i}}[c(x) = f(x)] \geq \frac{1}{2} + \frac{\gamma}{2}$, and therefore $\frac{Adv_c(M_i)}{\sum_x M_i(x)} \geq \gamma$ for $c = c_{i+1}$.

(We do not have enough time to continue the proof in Lecture 17, so the following texts provide a sketch of the proof.) We should always have $\sum_x M_i(x) \geq \varepsilon 2^n$ if the algorithm does not stop. This gives a lower bound on $\sum_x R_{c_{i+1}}(x) \cdot M_i(x) = Adv_{c_{i+1}}(M_i) \geq \gamma\varepsilon 2^n$. On the other hand, for any fixed input x, letting $A_i(x) := \sum_{0 \leq j \leq i-1} R_{c_{j+1}}(x) \cdot M_j(x)$, we should have $A_i(x) \leq \frac{1}{\varepsilon\gamma} + \frac{\varepsilon\gamma}{2} \cdot i$. (This is because, if $R_{c_{j+1}}(x)$ keeps being large for a few iterations j, then $M_i(x)$ drops very quickly by the definition.)

At last, we combine the upper and lower bounds, deduce a contradiction and therefore proving that the algorithm must terminate in T stages. Let us make this proof rigorous in the next section (which is covered in Lecture 18).

5 The Full Proof

Notice that $\sum_x M_i(x)$ captures the total measure of the space of $\{-1, 1\}^n$ that the next leaner (at stage $i + 1$) is trying to learn. If $\sum_x M_i(x) < \varepsilon 2^n$, the algorithm would have stopped. This is because, according to the definition, $\frac{1}{2^n}\sum_x M_i(x)$ upper bounds the fraction of the inputs x such that $Maj(c_1(x), c_2(x), \ldots, c_i(x)) \neq f(x)$. Therefore, we can safely assume that $\sum_x M_i(x) \geq \varepsilon 2^n$ for all $i \in \{0, 1, \ldots, T\}$; this gives a lower bound on the advantage

$$\sum_x R_{c_{i+1}}(x) \cdot M_i(x) = Adv_{c_{i+1}}(M_i) \geq \gamma\varepsilon 2^n \ .$$

For any fixed input x, letting $A_i(x) := \sum_{0 \leq j \leq i-1} R_{c_{j+1}}(x) \cdot M_j(x)$, we claim that

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

Let us first see how this claim implies our boosting theorem. Consider a matrix where the 2^n rows are all possible x's, and the columns are $0, 1, \ldots, i$, representing iterations. At the x-the row and the j-th column of the matrix, we put $R_{c_{j+1}}(x) \cdot M_j(x)$.

The sum of the x-th row of this matrix is $A_{i+1}(x)$. Therefore, owing to Claim 2, this gives an upper bound on the summation of all the entries of the matrix $\le 2^n\big(\frac{1}{\varepsilon\gamma} + \frac{\varepsilon\gamma}{2} \cdot (i+1)\big)$.

On the other hand, the sum of j-th column of this matrix is $\sum_x R_{c_{j+1}}(x) M_j(x) = Adv_{c_{j+1}}(M_j) \ge \gamma\varepsilon 2^n$. This gives a lower bound on the summation of all the entries of the matrix $\ge \gamma\varepsilon 2^n \cdot (i+1)$.

Together, we must have $2^n\big(\frac{1}{\varepsilon\gamma} + \frac{\varepsilon\gamma}{2} \cdot (i+1)\big) \ge \gamma\varepsilon 2^n \cdot (i+1)$, which implies that $i \le \frac{2}{\varepsilon^2\gamma^2} - 1$, and therefore the algorithm must terminate in $T = O(\frac{1}{\varepsilon^2\gamma^2})$ stages.

At last, we only need to prove the claim.

Proof [Proof of Claim 2] We will use an "elevator argument": in an elevator that starts from the first floor (and may have negative floors), no matter which floors we have visited, the number of times we ascend from floor k to $k+1$ is at most one more time than we descend from floor $k+1$ to k. This is also true for negative floors.

The proof of this elevator argument is very simple: for each pair of floors, say k and $k+1$, we can match each time we go up from k to $k+1$, to the most recent time we go down from $k+1$ to k. Then, for each pair of consecutive floors $k-(k+1)$, there is at most one extra "up" (if $k > 0$) or "down" (if $k \le 0$) that is left unmatched.

Now we go back to upper bound $A_i(x) := \sum_{0 \le j \le i-1} R_{c_{j+1}}(x) \cdot M_j(x)$. In this summation, there are i terms $j = 0, 1, \ldots, i-1$. For each (possibly negative) k, we match any $a = j$ satisfying $N_j(x) = k, N_{j+1}(x) = k+1$ (therefore is "going up"), to its neighboring $b = j'$ satisfying $N_{j'}(x) = k+1, N_{j'+1}(x) = k$ (therefore is "going down").

For any matched pair $a, b \in \{0, 1, \ldots, i-1\}$, the two corresponding terms sum up to

$$R_{c_{a+1}}(x)M_a(x) + R_{c_{b+1}}(x)M_b(x) = +1 \cdot M_a(x) - 1 \cdot M_b(x) = M_a(x) - M_b(x) \ . \tag{1}$$

This is because, a, by definition, is a "going up" term so we have $R_{c_{a+1}}(x) = N_{i+1}(x) - N_i(x) = +1$, and similarly $R_{c_{b+1}} = -1$. Next, if $k < 0$ is negative, we have $M_a(x) = M_b(x) = 1$ so (1) equals to zero; if $k > \frac{1}{\gamma\varepsilon}$ is large, we have $M_a(x) = M_b(x) = 0$ so (1) equals to zero; otherwise, $M_a(x) - M_b(x) = (1 - \varepsilon\gamma N_a(x)) - (1 - \varepsilon\gamma N_b(x)) = \varepsilon\gamma((-k) + (k+1)) = \varepsilon\gamma$. In sum, letting $P \subseteq \{0, 1, \ldots, i-1\}$ be the set of pairs that have matched, we have that their summation

$$\sum_{j \in P} R_{c_{j+1}}(x) \cdot M_j(x) \le \varepsilon\gamma \cdot \frac{|P|}{2} = \varepsilon\gamma \cdot \frac{i}{2} \ .$$

The unmatched coordinates $j \in \bar{P} := \{0, 1, \ldots, i-1\} \setminus P$, they must either have all $N_j(x) - N_{j+1}(x) = -1$ being negative (so "going down"), or have all $N_j(x) - N_{j+1}(x) = +1$ being positive (so "going up"). If it is the former "going up" case, the corresponding $R_{c_{j+1}}(x) = -1$ is negative but $M_j(x) \ge 0$ is always non-negative. This shows that the total sum $\sum_{j \in \bar{P}} R_{c_{j+1}}(x) \cdot M_j(x) \le 0$ is non-negative.

Otherwise, in the latter (so "going down") case, the corresponding $R_{c_{j+1}}(x) = 1$ is positive. For those $j \in \bar{P}$ that has $N_j(x) \ge \frac{1}{\gamma\varepsilon}$, they must satisfy $M_j(x) = 0$ so do not contribute to the summation; for those other $j \in \bar{P}$ with $N_j(x) \in \{0, 1, \ldots, \frac{1}{\gamma\varepsilon}\}$, they satisfy $M_j(x) \le 1$ so contributing 1 to the summation. In sum,

$$\sum_{j \in \bar{P}} R_{c_{j+1}}(x) \cdot M_j(x) \le \frac{1}{\varepsilon\gamma} \ .$$

Lastly, we conclude that $A_i(x) = \sum_{0 \le j \le i-1} R_{c_{j+1}}(x) \cdot M_j(x) \le \frac{1}{\varepsilon\gamma} + \varepsilon\gamma \cdot \frac{i}{2}$. ■