6.842 Randomness and Computation April 22, 2012

Lecture 19

Lecturer: Ronitt Rubinfeld *Scribe: Joshua Alman*

Today, we will:

- Finish with weak learning of monotone functions
- Start proving that weak learnability implies strong learnability

1 Learning Monotone Functions

Recall that if $x, y \in \{\pm 1\}^n$, then we write $x \leq y$ if for all coordinates $i \in [n]$, we have that $x_i \leq y_i$. Then:

Definition 1 (Monotone functions) *A function f is* monotone *if $f(x) \leq f(y)$ whenever $x \leq y$.*

We finish off our discussion of monotone functions from last lecture by proving:

Theorem 2 *For any monotone function $f : \{\pm 1\}^n \to \{\pm 1\}$, there is a $g \in \{+1, -1, \chi_{\{1\}}, \chi_{\{2\}}, \cdots, \chi_{\{n\}}\}$ such that*

$$\Pr_{x \in \{\pm 1\}^n} [f(x) = g(x)] \geq \frac{1}{2} + \Omega\left(\frac{1}{n}\right).$$

Proof If $\Pr_{x \in \{\pm 1\}^n} [f(x) = 1] > \frac{3}{4}$, then the function $g(x) = 1$ satisfies the desired probability, and if $\Pr_{x \in \{\pm 1\}^n} [f(x) = 1] < \frac{1}{4}$, then the function $g(x) = -1$ does. It remains to show the theorem for when

$$\Pr_{x \in \{\pm 1\}^n} [f(x) = 1] \in \left[\frac{1}{4}, \frac{3}{4}\right].$$

From problem 5 of problem set 3, we know that since f is monotone:

$$Inf_i(f) = \hat{f}(\{i\}).$$

Thus, if we can show that for some coordinate i, $Inf_i(f) = \Omega\left(\frac{1}{n}\right)$, then we will have that:

$$\Pr\left[f(x) = \chi_{\{i\}}(x)\right] = \frac{\hat{f}(\{i\}) + 1}{2} = \frac{1}{2} + \Omega\left(\frac{1}{n}\right),$$

as desired. Thus, we will show that there is an i such that $Inf_i(f) = \Omega\left(\frac{1}{n}\right)$ by using a different interpretation of $Inf_i(f)$.

In particular, we will use the hypercube graph interpretation of $Inf_i(f)$. Elements of $\{\pm 1\}^n$ are the vertices of the hypercube graph, and two vertices are adjacent if they have Hamming distance one. Last lecture, we colored the vertices: a vertex x is red if $f(x) = 1$, and blue if $f(x) = -1$. Then, we found that:

$$Inf_i(f) = \frac{\text{total \# red-blue edges in } i\text{th direction}}{\text{total \# edges in } i\text{th direction}} = \frac{\text{total \# red-blue edges in } i\text{th direction}}{2^{n-1}}$$

We will bound the number of red-blue edges in the i direction below by using a canonical paths.

Canonical Path Argument Plan

1) Define a canonical path for every pair (x, y) of (not necessarily adjacent) nodes such that x is red and y is blue.

2) Give an upper bound on the number of canonical paths passing through any edge of the hypercube.

3) Combine these to conclude a lower bound on the total number of red-blue edges, and thus the number of such edges in a particular direction i.

We begin with step 1:

Definition 3 (Canonical Path) *For all pairs (x, y) of nodes in the hyprcube, a* canonical path from x to y *scans bits from left to right, flipping bits where needed. Each flip corresponds to a step in the path.*

In this proof, we only care about canonical paths from x to y where x is red and y is blue, but the definition works in general.

For instance, if $x = -1, +1, +1, +1$ and $y = +1, -1, +1, -1$, then the canonical path between them is:

$$\begin{array}{rcccc} x = & \text{-1} & +1 & +1 & +1 \\ & \mathbf{+1} & +1 & +1 & +1 \\ & +1 & \mathbf{\text{-1}} & +1 & +1 \\ y = & +1 & \text{-1} & +1 & \mathbf{\text{-1}} \end{array}$$

Notice that x and y above are not comparable (it is neither true that $x \leq y$ nor $y \leq x$), but there is still a canonical path between them. Furthermore, it is clear that each step in a canonical path is an edge in the hypercube graph, and so this defines a path between any pair f vertices in the hypercube.

How many canonical paths from a red vertex to a blue vertex are there? Since there is exactly one canonical path between every pair of nodes, this is just the number of pairs of one red vertex and one blue vertex. Since $\Pr_{x \in \{\pm 1\}^n}[f(x) = 1] \in \left[\frac{1}{4}, \frac{3}{4}\right]$, the number of blue nodes is at least $\frac{1}{4}$ the total number of nodes, as is the number of red nodes. Thus, the number of pairs of a red node and a blue node is at least

$$\left(\frac{1}{4} \cdot 2^n\right)^2 = \frac{2^{2n}}{16} = 2^{2n-4}.$$

Next, for each edge of the hypercube graph, how many canonical paths cross it? Consider any edge $(w, w^{\oplus j})$ of the hypercube. If this is on a canonical path between x and y, then it must be the step that changes the jth bit. That means that the steps to change bits 1 through $j-1$ have to have happened before this edge, and so $w_1 = y_1$, $w_2 = y_2$, $\cdots$ $w_{j-1} = y_{j-1}$. Similarly, bits $j+1$ through n have not yet been changed on this canonical path, so they are still the same as x, meaning $w_{j+1} = x_{j+1}$, $\cdots$ $w_n = x_n$. Moreover, because flipping the jth bit of w is on the path, we know that $x_j = w_j$ and $y_j = -w_j$. The only bits of x and y that we have not accounted for are the first $j-1$ bits of x and the last $n-j-1$ bits of y. In fact, for any choice of these bits, w will be on the path from x to y. Thus, the total number of canonical paths containing the edge $(w, w^{\oplus j})$ is $2^{j-1} \cdot 2^{n-j-1} = 2^{n-1}$.

Finally, we can combine these two steps by noting that every path from a red node to a blue node must pass through at least one red-blue edge. Thus:

$$\#\text{ of red-blue edges} \geq \frac{\#\text{ of red-blue canonical paths}}{\max \#\text{ of canonical paths that cross an edge}} \geq \frac{2^{2n-4}}{2^{n-1}} = 2^{n-3},$$

and so by the Pigeonhole principle, there is a direction $i \in [n]$ such that

$$\#\text{ of red-blue edges in direction } i \geq \frac{1}{n} 2^{n-3}.$$

Thus, for that i:

$$Inf_i(f) \geq \frac{\frac{1}{n}2^{n-3}}{2^{n-1}} = \frac{1}{4n} = \Omega\left(\frac{1}{n}\right),$$

as desired. ■

2 Weak Learning

What we just showed is not that monotone functions are learnable by a PAC, but that they are "weakly learnable:"

Definition 4 (Weak Learning) *An algorithm $\mathscr{A}$* weakly PAC learns *concept class $\mathscr{C}$ if for all $c \in \mathscr{C}$ and all distributions $\mathscr{D}$ on $\mathscr{C}$, there exists a $\gamma > 0$ and $\delta = \frac{1}{4}$ such that with probabilitiy $\geq 1 - \delta$, given examples of c addording to $\mathscr{D}$, the algorithm $\mathscr{A}$ outputs an h such that* $\Pr_{\mathscr{D}}[h(x) = c(x)] \geq \frac{1}{2} + \frac{\gamma}{2}$.

A few notes about this definition, comparing it to the definition of (strong) learning from lecture 12:

- Here we set $\delta = \frac{1}{4}$ whereas in the definition of (strong) learning we pick any $\delta > 0$. This is because, like in problem 4 on problem set 3, the sample complexity of $\mathscr{A}$ could be made to be polynomial in $\log\left(\frac{1}{\delta}\right)$, which is relatively small, so we just treat δ as a constant to simplify the definition.
- γ should be thought of as a constant as well. We use $\frac{1}{2} + \frac{\gamma}{2}$ instead of $\frac{1}{2} + \gamma$ for convenience later; using either would yield the same definition.
- In (strong) learning, we have that for input ϵ, $\Pr_{\mathscr{D}}[h(x) = c(x)] \geq 1 - \epsilon$ rather than $\Pr_{\mathscr{D}}[h(x) = c(x)] \geq \frac{1}{2} + \frac{\gamma}{2}$ (where γ is fixed and not necessarily under the control of the user). This is the main difference between the two notions.

We put 'strong' in parentheses in '(strong) learning' because in lecture 12 we defined it to be called 'learning.' The word 'strong' just helps us to contrast the two terms.

It seems on first glance that weak learnability is a weaker concept than learnability. In fact, this was conjectured to be the case for many years. However, it turns out to be false.

Theorem 5 (Schapire) *Weak learnability implies (strong) learnability.*

We will give the proof of Klivans and Servedio which is based on the techniques of Impagliazzo. We will give a boosting method that converts a weak learning algorithm into a (strong) learning algorithm. However, this method is tricky, and we will not be able to cover it in its entire detail today.

2.1 An Intuitive Idea

Here is a first idea for a boosting method: Suppose a weaker learner is only 51% accurate. We can first learn a weak hypothesis, filter away examples which are correctly classified, and then call the weak learner on the remaining 49% of the data. To increase the collective coverage of the hypotheses, we can repeat this many times until we have a set of hypotheses that cover most of the data. Unfortunately, this method has a big problem: once our set of hypotheses is complete, then given an unseen example, which hypothesis shall we use to evaluate it at?

The basic idea of the boosting algorithm we give is to construct a filtering mechanism so that the majority vote of the collective hypotheses works out.

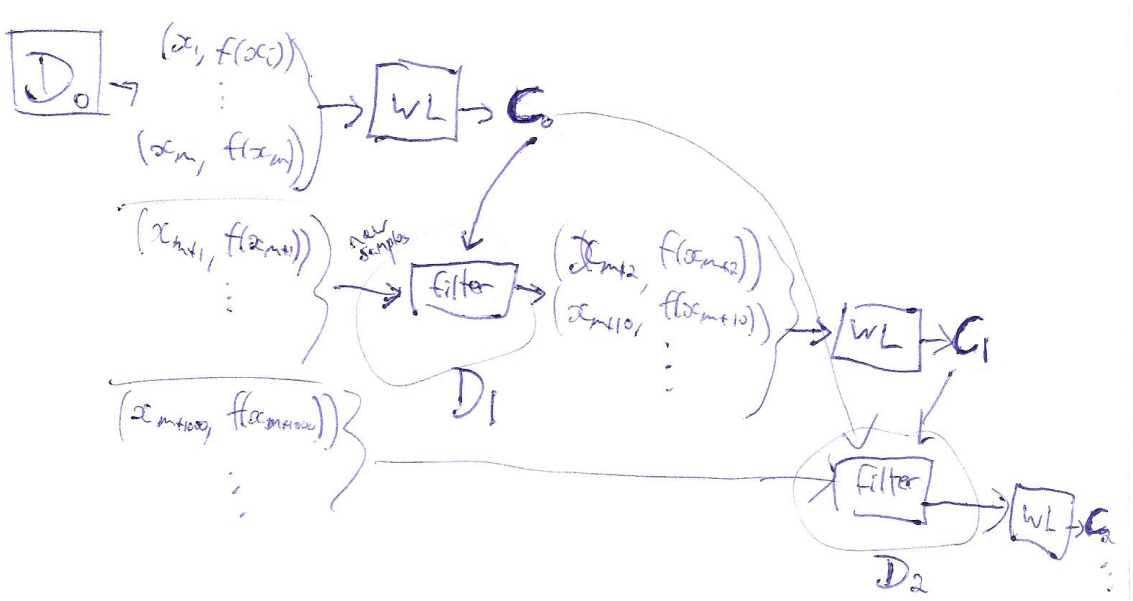
Proof We are trying to learn a function $f : \{\pm 1\}^n \to \{\pm 1\}$ from concept class $\mathscr{C}$. We are given a distribution $\mathscr{D}$ on $\mathscr{C}$ (we assume that $\mathscr{D}$ is the uniform distribution, althoguh this is not required for

the theorem), and an algorithm WL, which is a weak learning algorithm for $\mathscr{C}$ with parameter $\frac{\gamma}{2}$ for any distribution. We want to design a (strong) learning algorithm for $\mathscr{C}$ with parameter ϵ. Here is an iterative structure for our algorithm:

Stage 0: (initialize) Set $\mathscr{D}_0 = \mathscr{D}$. Then, use WL to generate, with high probability, a $c_0 \in \mathscr{C}$ such that $\Pr_{\mathscr{D}}[f(x) = c_0(x)] \geq \frac{1}{2} + \frac{\gamma}{2}$.

Stage i: At this point, we already have constructed $\mathscr{D}_0, \cdots, \mathscr{D}_{i-1}$ and $c_0, \cdots, c_{i-1}$. We then use a filtering procedure to construct $\mathscr{D}_i$. This filtering procedure will be fully described next lecture. We then run WL on $\mathscr{D}_i$ to output, with high probability, a c_i such that $\Pr_{\mathscr{D}}[f(x) = c_i(x)] \geq \frac{1}{2} + \frac{\gamma}{2}$.

After $T = O\left(\frac{1}{\gamma^2 \epsilon^2}\right)$ stages: output $c = MAJ(c_1, c_2, \cdots, c_T)$.



We will describe in more detail how the filter works and complete the proof next lecture. ■