6.842 Randomness and Computation April 7, 2014

Lecture 16

Lecturer: Ronitt Rubinfeld *Scribe: James Weis*

Covered in this lecture:

- Fast weak learning of monotone functions
- Begin discussion of "weak learnability implies strong learnability"

1 Learning Monotone Functions

Recall the following definitions:

Definition 1 (Partial order, $\preceq$) *If $x, y \in \{\pm 1\}^n$, then we write $x \preceq y$ if for all coordinates $i \in [n]$, we have that $x_i \leq y_i$.*

Definition 2 (Monotone functions) *A function f is* monotone *if $f(x) \leq f(y)$ whenever $x \leq y$.*

We begin by proving the following:

Theorem 3 *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 We have two cases; either $f(x)$ has weak agreement with ± 1, or $\Pr[f(x) = 1] \in \left[\frac{1}{4}, \frac{3}{4}\right]$.

Regarding the first case, 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 $\Pr_{x \in \{\pm 1\}^n}[f(x) = 1] \in \left[\frac{1}{4}, \frac{3}{4}\right]$, then the function $g(x) = -1$ does.

It remains to show the second case; namely, that the theorem holds when

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

Recall the following theorem from the homework:

Theorem 4 *For f monotone, $Inf_i(f) = \hat{f}(\{i\}) \equiv 2\Pr\left[f(x) = \chi_{\{i\}}(x)\right] - 1$.*

Our plan is then to show that, for some coordinate i, $Inf_i(f) \leq \Omega\left(\frac{1}{n}\right)$, 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. Previously, 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 the following tool:

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 (note that such a path *must* cross at least one red-blue edge).

2) Give an upper bound on the number of canonical paths passing through $\underline{\text{any}}$ edge of the hypercube (in paticular, any red-blue edge).

3) Combine these two 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 5 (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:

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

Notice that x and y above are not comparable (it is neither true that $x \preceq y$ nor $y \preceq x$), 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 of vertices in the hypercube.

For the first part of our plan, we want to determine how many canonical paths from a red vertex to a blue vertex there are. 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 $\Pr_{x\in\{\pm1\}^n}[f(x) = 1] \in \left[\frac{1}{4}, \frac{3}{4}\right]$, 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

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

We now attend to the second part of our plan—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 combine these two conclusions by noting that every path from a red node to a blue node must pass through at least one red-blue edge. We have that

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

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

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

And so, 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" vs "Strong" Learning

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

Definition 6 (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 $\Pr_{\mathscr{D}}[h(x) = c(x)] \geq \frac{1}{2} + \frac{\gamma}{2}$.*

We make a few notes about this definition by comparing it to "strong" learning:

- Here we set $\delta = \frac{1}{4}$ whereas in the definition of strong learning we pick any $\delta > 0$. This is because 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.

Note that our previous definition of learning was really "strong" learning. However, now that we have "weak" learning, we use the word "strong" 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, as it is possible to "boost" a weak learner (note that one could also simply simulate a weak learner several times on some distribution and take the majority or best answer. This would provide increased confidence, but would not reduce error).

Theorem 7 (Schapire) *If any concept class $\mathscr{C}$ can be weakly learned on* any *distribution $\mathscr{D}$, then $\mathscr{C}$ can be "strongly" learned.*

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