6.842 Randomness and Computation April 20, 2022

Lecture 21

Lecturer: Ronitt Rubinfeld *Scribe: Jacob Teo*

In this lecture, we will cover weak learning of monotone functions.

1 Review

Definition 1 *A function $f : X \to Y$ is* ***monotone*** *if, given a partial order $\leq$ on X, for any $x, y \in X$, $x \leq y \implies f(x) \leq f(y)$.*

We define a partial order $\leq$ on the elements of $\{-1,1\}^n$ as follows: For any $x, y \in \{-1,1\}^n$, $x \leq y$ if and only if $\forall i x_i \leq y_i$. We can imagine this as a graph on vertices $\{-1,1\}^n$, where we have an edge from A to B if we can change a single -1 to 1 in A to get B. This is essentially a directed version of the Boolean hypercube graph.

For every node in the graph, we define its **level** as the number of bits it has equal to 1. Thus, in level k there are $\binom{n}{k}$ nodes. Note that every edge must go from some level i to $i+1$.

Definition 2 *A function $f : \{-1,1\}^n \to \{-1,1\}$, for even n, is a* ***slice function*** *if $f(x) = -1$ if level$(x) < n/2$ and $f(x) = 1$ if level$(x) > n/2$. The value of $f(x)$ when level$(x) = n/2$ is arbitrary.*

Corollary 3 *Every slice function is a monotone function.*

How big is the class of slice functions? Notice that all assignments of nodes in level $n/2$ are valid slice functions, and hence there are $2^{\binom{n}{n/2}} \approx 2^{\frac{2^n}{\sqrt{n}}}$ possible slice functions.

Question: Is there a fast learning algorithm specifically for **monotone functions**?

First let us restrict our class to slice functions, which are all monotone. By Occam's razor, the number of samples we need is of the order $\log 2^{\binom{n}{n/2}} = \tilde{O}(\frac{2^n}{\sqrt{n}})$ which is very large.

However, notice that the majority function $g(x) = \text{sign}(\sum_{i=1}^n x_i)$ is correct on all inputs x where level$(x) \neq n/2$, and hence is guaranteed to be correct for all except $\approx \frac{1}{\sqrt{n}}$ of inputs, and is thus a good approximation with respect to the uniform distribution with essentially zero samples, which gives a much better bound than Occam's razor, specifically for the class of slice functions. This demonstrates how Occam's razor does not necessarily give us the best bound.

2 Weak Learning for Monotone Functions

We first try to find a weak learning algorithm for monotone functions. Weak learning yields an algorithm that is only *slightly* better than random guessing (hence the term "weak"). We present a theorem that shows that all monotone functions have *weak agreement* with some dictator or constant function.

Theorem 4 *For all monotone functions f, $\exists g \in \{\pm 1, x_1, x_2, ...x_n\} = \mathcal{S}$ such that* $\Pr_x[f(x) = g(x)] \geq \frac{1}{2} + \Omega(\frac{1}{n})$.

In other words, for any monotone function f, there exists a function g that is either *constant* or a *dictator function*, such that g approximates f slightly better than random guessing.

This theorem also implies a simple algorithm to find such a function g: simply try all members of $\mathcal{S}$ and output the best one via sampling.

To prove this theorem, we split it into two cases. Suppose that $f(x)$ has weak agreement with $g(x) = +1$ or -1, then we are done. Otherwise, $\Pr[f(x) = 1]$ must be very close to $1/2$, and we can bound it as $\Pr[f(x) = 1] \in [1/4, 3/4]$ (note that this is an extremely loose bound). Before continuing, we first introduce a new general technique for proving this theorem.

3 Canonical Path Argument

First, we consider how we can view a monotone function f as a coloring on the directed Boolean hypercube graph. We color each node **red** if $f(x) = 1$ and **blue** if $f(x) = -1$. Then monotonicity implies that if we take any path from $\{-1\}^n$ to $\{1\}^n$, there will be some edge on the path where the nodes turn from blue to red.

The blue-red edges form a "boundary" between blue and red in the colored graph.

Definition 5 *The **influence** of f with respect to the i-th bit is defined by:*

$$\mathrm{Inf}_i(f) = \frac{\#\ \textit{red-blue edges in } i\textit{-th direction}}{2^{n-1}} = \Pr[f(x) \neq f(x^{\oplus i})]$$

where ($x^{\oplus i}$ denotes x with the i-th bit flipped.

*The overall **influence** of f is the sum over all i:*

$$\mathrm{Inf}(f) = \frac{\#\ \textit{red-blue edges}}{2^n} = \sum_{i=1}^{n} Inf_i(f)$$

We now present two theorems, the proofs of which will be in the homework.

Theorem 6 *For a monotone function f, $\mathrm{Inf}_i(f) = \hat{f}(\{i\})$.*

Theorem 7 *For odd n, the majority function $f(x) = \mathrm{sign}(\sum_{i=1}^n x_i)$ has the maximal influence among all monotone functions.*

Now recall that

$$\hat{f}(\{i\}) = 2 \cdot \Pr[f(x) = X_{\{i\}}(x)] - 1 = 2 \cdot \Pr[f(x) = x_i] - 1$$
$$\iff \Pr[f(x) = x_i] = \frac{1}{2} + \frac{\mathrm{Inf}_i(f)}{2}$$

and hence to prove our theorem, it suffices to find an i such that $\mathrm{Inf}_i(f) \geq \Omega(\frac{1}{n})$.
To show that this i exists, we use a technique called the **Canonical Path Argument**. The structure of the proof will be as follows:

1. For every red-blue pair of nodes, we define a *canonical path* (a deterministic, unique path). Note that this path must cross at least one red-blue edge.
2. Next we upper bound the number of canonical paths passing through a given red-blue edge e.
3. Then knowing the total number of canonical paths, we have a lower bound on the total number of red-blue edges.

3.1 Defining the canonical path

Definition 8 *For every pair of nodes x, y where x is red and y is blue, we define the* ***canonical path*** *from x to y as follows: We scan the bits from left to right ($i = 1$ to n) then flip the bit if needed (if $x_i \neq y_i$). Each flip corresponds to traversing a single edge and is a step in the path.*

3.2 Counting canonical paths

Each red-blue pair has a corresponding canonical path. Recall that we have $\Pr[f(x) = 1] \in [\frac{1}{4}, \frac{3}{4}]$, and hence at least $\frac{1}{4}$ of the 2^n nodes are red and $\frac{1}{4}$ are blue. Thus

$$\text{Number of paths } (\geq \frac{1}{4} \cdot 2^n) \cdot (\frac{1}{4} \cdot 2^n) = \frac{1}{16} 2^{2n}$$

3.3 Bounding paths through any edge

Each edge corresponds to flipping some bit i. Let us denote the edge as $(u, u^{\oplus i})$. Then we want to count x, y such that the canonical path of $x \to y$ will pass through $u \to u^{\oplus i}$. To achieve this, x must agree with u on bits $i, i+2, ...n$ and y must agree with $u^{\oplus i}$ on bits $1, 2, ...i$. Thus we have 2^{i-1} ways to choose $x_1, x_2, ...x_{i-1}$ and 2^{n-i} ways to choose $y_{i+1}, y_{i+2}, ...y_n$. Hence the total number of x, y pairs is $2^{i-1} \cdot 2^{n-i} \leq 2^n$.

3.4 Final bound

Recall that every canonical path must pass through at least one red-blue edge. Hence

$$\text{Number of red-blue edges } \times \text{ Max number of canonical paths passing through an edge}$$

$$\geq \text{Number of red-blue canonical paths}$$

$$\implies \text{Number of red-blue edges} \geq \frac{\frac{1}{16} \cdot 2^{2n}}{2^n} = \frac{1}{16} \cdot 2^n$$

Every edge is in one of n directions, and hence

$$\exists i \text{ such that there are } \geq \frac{1}{16} \cdot 2^n \cdot \frac{1}{n} \text{ red-blue edges in direction } i$$

Since we know each direction i has 2^{n-1} edges,

$$\implies \exists i \text{ such that } \mathrm{Inf}_i(f) \geq \frac{\frac{1}{16} \cdot 2^n \cdot \frac{1}{n}}{2^{n-1}} = \frac{1}{8n} = \hat{f}(\{i\}) = 2 \cdot \Pr[f(x) = x_i] - 1$$

$$\implies \exists i \text{ such that } \Pr[f(x) = x_i] \geq \frac{1}{2} + \frac{1}{16n}$$

and this i completes the proof of our theorem.

4 Weak Learning

We now discuss weak learning as a general concept.

Definition 9 *An algorithm $\mathcal{A}$* ***weakly PAC learns*** *concept class $\mathcal{C}$ if $\forall c \in \mathcal{C}$ and over all distributions $\mathcal{D}$, there exists $\gamma > 0$ such that $\forall \delta > 0$ with probability $\geq 1 - \delta$ given examples of c, $\mathcal{A}$ outputs h such that* $\Pr_{\mathcal{D}}[h(x) = c(x)] \geq \frac{1}{2} + \frac{\gamma}{2}$.

Notice that the final term is changed from $1 - \epsilon$ in strong learning to $\frac{1}{2} + \frac{\gamma}{2}$ in weak learning.

It was first conjectured that weak learning is easier that weak learning, which meant that there exists functions that we can weakly learn but not strongly learn. The next theorem disproves that, through a technique called "boosting".

Theorem 10 *If $\mathcal{C}$ can be weakly learned on any distribution $\mathcal{D}$, then $\mathcal{C}$ can be strongly learned.*

We will continue discussing this in the next lecture.