Lecture 18

Lecturer: Ronitt Rubinfeld *Scribe: Sophia Kwon*

Today, we continue on the topic of Fourier-based learning algorithms. We will finish our analysis of the low degree algorithm from last time, and we'll continue on to cover Fourier concentration and noise sensitivity:

1. Low degree algorithm
2. Fourier concentration
3. Noise sensitivity

1 Low Degree Algorithm

1.1 Review of Fourier Transform

In previous lectures, we described a way to construct a basis to describe all possible functions f which take an n-bit input and produce a one-bit answer. (Recall the convention of using bits $\{+1, -1\}$ instead of $\{0, 1\}$.) We used **parity functions**: for $S \subseteq \{1, ..., n\}$ and $x \in \{\pm 1\}^n$, we define the parity function

$$\chi_S(x) = \prod_{i \in S} x_i$$

In addition, we define the **normalized inner product**

$$\langle f, g \rangle = \frac{1}{2^n} \sum_{x \in \{\pm 1\}^n} f(x) g(x)$$

We proved that the set of parity functions $\{\chi_S\}$ is an **orthonormal basis** with respect to the normalized inner product.

Because the set of parity functions $\{\chi_S\}$ is an orthonormal basis, any function f is uniquely expressible as a linear combination of χ_S. We defined the **Fourier coefficients of f** as $\{\hat{f}(S)\}$ where

$$\begin{aligned} \hat{f}(S) &\equiv \langle f, \chi_S \rangle \\ &= \frac{1}{2^n} \sum_{x \in \{\pm 1\}^n} f(x) \chi_S(x) \end{aligned}$$

And we proved that for all f, $f(x)$ can be expressed as a linear combination of the parity functions, with the Fourier coefficients being the coefficients of the linear combination:

$$\forall f, f(x) = \sum_S \hat{f}(S) \chi_S(x)$$

Fourier coefficients also characterize the *distance to linearity* of a function. We showed that

$$\hat{f}(S) = 1 - 2 \cdot Pr_{x \in \{\pm 1\}^n}[f(x) \neq \chi_S(x)]$$

We also covered the useful equality **Plancherel's identity**:

$$\langle f, g \rangle = \sum_S \hat{f}(S) \hat{g}(S)$$

1.2 Learning via Fourier Representation

With that in mind, we turned our attention to learning algorithms based on estimating the Fourier representation of a function f. Last time we showed that we can approximate *one* Fourier coefficient.

Lemma 1 *For any $S \subseteq [n]$, you can approximate $\hat{f}(S)$ to within γ additive error (that is, $|output - \hat{f}(S)| \leq \gamma$) with probability $\geq 1 - \delta$ in $O(\frac{1}{\gamma^2} \log \frac{1}{\delta})$ samples.*

1.3 Low Degree Fourier Coefficients

What functions can we describe "pretty well" using low degree Fourier coefficients (corresponding to small $|S|$)? To answer that, we introduce the idea of **Fourier concentration**.

Definition 1 *A function $f : \{\pm 1\}^n \rightarrow \mathbb{R}$ has $\alpha(\epsilon, n)$-**Fourier concentration** if $\forall 0 < \epsilon < 1$,*

$$\sum_{S \subseteq [n] : |S| > \alpha(\epsilon, n)} \hat{f}(S)^2 \leq \epsilon$$

For boolean functions f, this implies

$$\sum_{S \subseteq [n] : |S| \leq \alpha(\epsilon, n)} \hat{f}(S)^2 \geq 1 - \epsilon$$

1.4 The Low Degree Algorithm

The **low degree algorithm** approximates functions with $d \equiv \alpha(\epsilon, n)$-Fourier concentration. For a given degree d, accuracy τ, and confidence δ, the algorithm runs as follows:

- Take $m = O(\frac{n^d}{\tau} \ln \frac{n^d}{\delta})$ samples
- For each S such that $|S| \leq d$:
  - Let C_S be your estimate of $\hat{f}(S)$
- Let $h(X) \equiv \sum_{|S| \leq d} C_S \cdot \chi_S(x)$
- Output $sign(h)$ as hypothesis

Why does this work? We prove correctness in two stages:

1. We will show that if f has as low Fourier concentration, then the expected value of the normalized L_2-distance $E_x[(f(x) - h(x))^2]$ is small.
2. We will show that $Pr[f(x) \neq sign(h(x))] \leq E_x[(f(x) - h(x))^2]$

When we put these two results together, we will be able to conclude that if f as a low Fourier concentration, then f and $sign(h)$ disagree on only a few values of x, so $sign(h(x))$ is a good approximation of $f(x)$.

In the previous lecture, we addressed the first stage by proving the following theorem:

Theorem 1 *If f has $d \equiv \alpha(\epsilon, n)$-Fourier concentration, then h satisfies $E_x[(f(x) - h(x))^2] \leq \epsilon + \tau$ with probability $\geq 1 - \delta$.*

Now, we will take care of the second stage with the following theorem:

Theorem 2 *For $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$ and $h : \{\pm 1\}^n \rightarrow \mathbb{R}$,*

$$Pr_x[f(x) \neq sign(h(x))] \leq E_x[(f(x) - h(x))^2]$$

Here's the proof: By the definition of probability over values of x, we can say that the left hand side of the inequality in the theorem is

$$Pr_x[f(x) \neq sign(h(x))] = \frac{1}{2^n} \sum_x 1_{f(x) \neq sign(h(x))}$$

From the definition of expected value, we can say that the right hand side of the inequality in the theorem is

$$E_x[(f(x) - h(x))^2] = \frac{1}{2^n} \sum_x (f(x) - h(x))^2$$

Since both sides are $\frac{1}{2^n}$ times a sum of values over all x, we can compare corresponding terms for each x. If every left hand side term is less than or equal to its corresponding right hand side term, that is, $1_{f(x) \neq sign(h(x))} \leq (f(x) - h(x))^2$ for all x, then the inequality is true.

Each value of x falls into one of two cases:

Case 1 $f(x) = sign(h(x))$: In this case, the left hand side term $1_{f(x) \neq sign(h(x))} = 0$. The right hand side term $(f(x) - h(x))^2 \geq 0$ because the square of a real number is always non-negative. Therefore, $1_{f(x) \neq sign(h(x))} \leq (f(x) - h(x))^2$, so we're good.

Case 2 $f(x) \neq sign(h(x))$: In this case, $1_{f(x) \neq sign(h(x))} = 1$. As for the right hand side, we know that $f(x)$ and $h(x)$ have different signs. Recall that $f(x)$ is either $+1$ or -1. If $f(x) = +1$, then $h(x) < 0$, so $f(x) - h(x) > 1$, which means $(f(x) - h(x))^2 \geq 1$. If $f(x) = -1$, then $h(x) > 0$, so $f(x) - h(x) < -1$, which means that, again, $(f(x) - h(x))^2 \geq 1$. Thus, $1_{f(x) \neq sign(h(x))} \leq (f(x) - h(x))^2$.

Thus, for all x, $1_{f(x) \neq sign(h(x))} \leq (f(x) - h(x))^2$. So,

$$\frac{1}{2^n} \sum_x 1_{f(x) \neq sign(h(x))} \leq \frac{1}{2^n} \sum_x (f(x) - h(x))^2$$
$$Pr_x[f(x) \neq sign(h(x))] \leq E_x[(f(x) - h(x))^2]$$

■

1.5 Correctness of Learning Algorithm

We summarize our results so far in a theorem about the learnability of a concept class C with a certain Fourier concentration.

Theorem 3 *If concept class C has Fourier concentration $d = \alpha(\epsilon, n)$, then there is a $q = O(\frac{n^d}{\epsilon} \log \frac{n^d}{\delta})$ sample uniform distribution learning algorithm for C. In other words, there exists an algorithm which takes q samples and with probability $\geq 1 - \delta$ outputs h' such that $Pr_x[f(x) \neq h'(x)] \leq 2\epsilon$.*

Here's the proof: We can run the Low Degree Algorithm with $\tau = \epsilon$. By Theorem 1, the Low Degree Algorithm obtains an h such that the expected L_2 difference between f and h is

$$E_x[(f(x) - h(x))^2] \leq \epsilon + \epsilon = 2\epsilon$$

The algorithm outputs $h' \equiv sign(h)$. Theorem 2 implies that

$$Pr_x[f(x) \neq sign(h(x))] \leq 2\epsilon$$

■

2 Fourier Concentration

Now, we explore some applications of the Low Degree Algorithm.

2.1 Bounded-Depth Decision Trees

Recall from last lecture that in a decision tree, we define V_l as the set of variables visited on the path to leaf l. We define the path functions $f_l(x)$ as

$$\begin{aligned} f_l(x) &= \frac{1}{2^{|V_l|}} \sum_{S \subseteq V_l} (\pm 1) \cdot \chi_S(x) \\ &= \begin{cases} 1 \text{ if } x \text{ takes the path to } l \\ 0 \text{ otherwise} \end{cases} \end{aligned}$$

In a decision tree T,

$$f(x) = \sum_{l \in \text{ leaves of } T} f_l(x) \cdot val(l),$$

where $val(l)$ is the output value at leaf l.

By our definition of f_l, the number of variables that any given $f_l(x)$ depends on is at most the depth of the tree. And $val(l)$ is a constant. By the linearity of Fourier, we have

$$\hat{f}(S) = \sum val(l) \cdot \hat{f}_l(S)$$

This means that for all S which have size greater than the depth of the tree ($|S| >$ depth), $\hat{f}(S) = 0$. So f has *depth*-Fourier concentration. Therefore, by Theorem 3, we can use $O(\frac{n^{depth}}{\epsilon} \log \frac{n^{depth}}{\delta})$ samples to approximate f.

2.2 Constant Depth Circuits

We can think of any boolean circuit C as a directed acyclic graph where each node is a gate, which can be an operation ("AND" $\wedge$, "OR" $\vee$, or "NOT" $\neg$), a constant (1 or 0), or a variable $(x_1, ..., x_n)$. How many inputs are we allowed to wire into each $\wedge$ or $\vee$ gate? The answer depends on the model we use: some models allow for only a constant number of inputs to each gate (e.g. 2), some allow for a polynomial number of inputs to each gate. In our model, we will allow an *unbounded* number of inputs to each gate because we would like to observe behavior at the most "extreme" case.

Our question is: can we compute parity (XOR) of n bits in a circuit of constant depth? The answer is yes, we can use Karnaugh maps to compute any function on n bits in constant depth! But can we compute parity of n bits in a circuit of constant depth and size that is polynomial with respect to n? No, according to the switching lemma proved by Furst, Saxe, and Sipser. However, we can use the Low Degree Algorithm to approximate the parity function $f(x)$ using a pseudo-polynomial number of samples.

Theorem 4 (Hastad, Linial Mansour Nisan) *For all functions f which are computable by circuits of size s and depth d,*

$$\sum_{|S|>t} \hat{f}^2(S) \leq \alpha$$

for $t = O(\log \frac{s}{\alpha})^{d-1}$.

It follows that any such f has Fourier concentration t. If the circuit size s is polynomial with respect to n, the circuit depth d is constant, and α is $O(\epsilon)$, then $t = O(\log^d(\frac{n}{\epsilon}))$. According to Theorem 3, this yields an algorithm which takes $n^{O(\log^d(\frac{n}{\epsilon}))}$ samples. Jackson showed that you can improve the algorithm to use $n^{O(\log\log n)}$ samples. (Recall that the parity of S will have one large Fourier coefficient of degree $|S|$.)

2.3 Learning Halfspaces

Definition 2 *$h(x) = sign(w \cdot x - \theta)$ is a **halfspace function**.*

(Recall that $sign(y) = +1$ if $y \geq 0$ and -1 otherwise.)

Theorem 5 *Let h be a halfspace over $\{\pm 1\}^n$. Then h has Fourier concentration $\alpha(\epsilon) = \frac{c}{\epsilon^2}$. That is,*

$$\sum_{|S| \geq c/\epsilon^2} \hat{h}(S)^2 \leq \epsilon$$

We will prove this later, but it leads us to the following corollary:

Corollary 1 *The Low Degree Algorithm learns halfspaces under a uniform distribution with $n^{O(1/\epsilon^2)}$ uniformly generated samples.*

3 Noise Sensitivity

We introduce the concept of **noise sensitivity**, which is used to bound Fourier concentration.

Definition 3 *A **noise operator** is the function $N_\epsilon(x) = x$ but with each bit randomly flipped with probability ϵ, where $0 < \epsilon < \frac{1}{2}$.*

Definition 4 ***Noise sensitivity** is how likely a function f changes if noise is added to its input x:*

$$NS_\epsilon(f) = Pr_{x \in \{\pm 1\}^n \text{ \& } noise}[f(x) \neq f(N_\epsilon(x))]$$

We give the noise sensitivity of several example functions in the sections below:

3.1 $f(x) = x_1$

The noise operator $N_\epsilon(x)$ flips x_1 with probability ϵ. Therefore,

$$\begin{aligned} NS_\epsilon(f) &= Pr[f(x) \neq f(N_\epsilon(x))] \\ &= Pr[N_\epsilon(x) \text{ flips } x_1] \\ &= \epsilon \end{aligned}$$

3.2 $f(x) = x_1 x_2 ... x_k$

$$\begin{aligned} NS_\epsilon(f) &= Pr[f(x) = \text{False} \wedge f(N_\epsilon(x)) = \text{True}] + Pr[f(x) = \text{True} \wedge f(N_\epsilon(x)) = \text{False}] \\ &= 2 \cdot Pr[f(x) = \text{False} \wedge f(N_\epsilon(x)) = \text{True}] \\ &= 2 \cdot \frac{1}{2^k}(1 - (1-\epsilon)^k) \end{aligned}$$

If $\epsilon << \frac{1}{k}$, then $NS_\epsilon(f)$ is approximately $\frac{1}{2^{k-1}}(\epsilon k)$.
If $\epsilon >> \frac{1}{k}$, then $NS_\epsilon(f)$ is approximately $\frac{1}{2^{k-1}}(1 - e^{-k\epsilon})$.

3.3 $f(x) = Maj(x_1, ..., x_n)$

$$NS_\epsilon(f) = O(\sqrt{\epsilon})$$

We'll just give a sketch for this result: You can simulate $Maj(x)$ using a random walk on a line. You start at 0. Every time you see a $+1$ input bit, you move right one. Every time you see a -1 input bit, you move left one. The value of $Maj(x)$ is the sign of the node you end up at. For example, on $x = (+1, -1, -1, +1, +1, +1)$, you would start at 0, move right to 1, move left to 0, move left to -1, move right to 0, move right to 1, and move right to 2. Since you end on a positive node (2), $Maj(x) = +1$. Note that this is equivalent to taking the sign of the sum of the input bits.

Then $N_\epsilon(x)$ is analogous to taking a random walk on a line of ϵn nodes. Each bit flip displaces our walk by ± 2 nodes (flipping -1 to +1 moves you to the right by two, and flipping +1 to -1 moves you left two).

Fact $E[|x_1 + x_2 + ... + x_n|] = \sqrt{n}$ and is likely to be close to $\sqrt{n}$.
By this fact, we know that the expected resulting displacement of our walk is

$$E[\text{displacement}] = 2\sqrt{\epsilon n}$$

So our process for determining whether $f(x) \neq f(N_\epsilon(x))$ will go as follows:

1. Take the walk specified by x.
2. Continue the walk according to $2 \cdot N_\epsilon(x)$.

Using a heuristic argument, we can pretend that the first walk leaves us at $\sqrt{n}$. $f(x) \neq f(N_\epsilon(x))$ if the second walk takes us across node 0. We can bound the probability this will happen:

$$Pr[2^{nd} \text{ walk takes us across } 0] = \frac{1}{2} Pr[2^{nd} \text{ displacement } > \sqrt{n}]$$

$\sqrt{n} = \frac{1}{2\sqrt{\epsilon}} \cdot 2\sqrt{\epsilon n}$, so by Markov's inequality, we have

$$Pr[2^{nd} \text{ walk takes us across } 0] \leq 2\sqrt{\epsilon}$$

So the majority function has noise sensitivity $\leq 2\sqrt{\epsilon}$.

3.4 Linear Threshold Function (Halfspace)

Theorem 6 (Peres) $NS_\epsilon(LTF) < 8.8\sqrt{\epsilon}$, *where* LTF *is any linear threshold function.*

Note that this is the best possible, since $NS_\epsilon(Maj) = \Theta(\sqrt{\epsilon})$.

3.5 Parity functions $\chi_S(x)$ for $|S| = k$

$$\begin{aligned} NS_\epsilon(f) &= Pr[N_\epsilon(x) \text{ flips an odd number of bits in } S] \\ &= \frac{1 - (1 - 2\epsilon)^k}{2} \end{aligned}$$