6.842 Randomness and Computation April 11, 2022

Lecture 19

Lecturer: Ronitt Rubinfeld *Scribe: Siwakorn Fuangkawinsombut*

In this lecture, we continue on Fourier-based learning algorithms and how we use the noise sensitivity to bound the Fourier concentration, as well as applying the technique derived from the noise sensitivity to learn halfspaces. Then, in the second half of the lecture, we shift to the topic of learning heavy Fourier coefficients with queries, which will mainly focus on the Kushilevitz-Mansour learning algorithm. Based on this, the notes are structured as follows:

1. Review
2. Fourier concentration via noise sensitivity
3. Learning heavy Fourier coeffs (with queries)

1 Review

1.1 Fourier Transform

Definition 1 *For $S \in [n]$ and $x \in \{\pm 1\}^n$, the **parity function** is*

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

Definition 2 *For any function f, the **Fourier coefficients** of f are $\{\hat{f}(S)\}$, where*

$$\hat{f}(S) = \langle f, \chi_S \rangle = 1 - 2Pr_x\left[f(x) \neq \chi_S(x)\right] = 2Pr_x\left[f(x) = \chi_S(x)\right] - 1.$$

A useful equation that we will use many times in this lecture, **Parseval's identity**:

$$\langle f, f \rangle = \sum_{S \subseteq [n]} \hat{f}(S)^2,$$

which is equal to one when f is a boolean function.

1.2 Noise Sensitivity

Definition 3 *A **noise operator** is the function $N_\epsilon(x)$ that each bit of x randomly flipped with probability ϵ, where $0 < \epsilon < 1/2$.*

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

$$NS_\epsilon(x) = Pr_{x \in \{\pm 1\}^n + noise}\left[f(x) \neq f(N_\epsilon(x))\right].$$

Theorem 5 *For any boolean function $f : \{\pm 1\}^n \to \{\pm 1\}$, the noise sensitivity of this function is*

$$NS_\epsilon(f) = \frac{1}{2} - \frac{1}{2} \sum_{S \subseteq [n]} (1 - 2\epsilon)^{|S|} \hat{f}(S)^2.$$

The proof of this theorem is left as an exercise and may be included in the next problem set.

2 Fourier Concentration via Noise Sensitivity

We introduced noise sensitivity in the last lecture. Now, we can utilize this concept to bound Fourier concentration. Thus, we can determine a nice Fourier concentration via noise sensitivity. Also, noise sensitivity provides a technique to be used to describe halfspaces by the corollary of the following inequality theorem.

Theorem 6 *For a boolean function f and probability $0 < \gamma < 1/2$, then*

$$\sum_{|S| \geq \frac{1}{\gamma}} \hat{f}(S)^2 < 2.32\, NS_\gamma(f).$$

Proof By theorem 5, we get

$$\begin{aligned} 2\, NS_\gamma(f) &= 1 - \sum_{S \subseteq [n]} (1-2\gamma)^{|S|} \hat{f}(S)^2 \\ &= \sum_{S \subseteq [n]} \left(1 - (1-2\gamma)^{|S|}\right) \hat{f}(S)^2 \\ &\geq \sum_{|S| \geq \frac{1}{\gamma}} \left(1 - (1-2\gamma)^{|S|}\right) \hat{f}(S)^2 \\ &> \sum_{|S| \geq \frac{1}{\gamma}} \left(1 - e^{-2}\right) \hat{f}(S)^2. \end{aligned}$$

The second line inequality is followed from the Parseval's identity. Thus, we have

$$\sum_{|S| \geq \frac{1}{\gamma}} \hat{f}(S)^2 < \left(\frac{2}{1-e^{-2}}\right) NS_\gamma(f),$$

where $\frac{2}{1-e^{-2}} \approx 2.31 < 2.32$ as desired. ■

If $NS_\gamma(f)$ is small enough, we can apply the low degree algorithm to $1/\gamma$ to find Fourier concentration. Now, we can use this theorem for learning halfapces as the follow corollary:

Corollary 7 *For a halfspace h, then*

$$\sum_{|S| \geq O\left(\frac{1}{\epsilon^2}\right)} \hat{h}(S) < \epsilon.$$

We can show this corollary by bounding on the noise sensitivity and some calculations. Hence, we can learn any halfspace from $n^{O\left(1/\epsilon^2\right)}$ random examples.

Corollary 8 *Any function of k halfspaces can be learned with $n^{O\left(k^2/\epsilon^2\right)}$ samples*

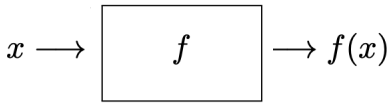
As we don't get examples from all k halfspaces but only from functions, we can't learn each of them individually. However, we can bound noise sensitivity!

3 Learning Heavy Fourier Coeffs (with queries)

Consider the scenario when we don't have many heavy, like five, but most big Fourier coefficients corresponding to big S's, the low degree algorithm won't work anymore. Thus, we need to query inputs for getting progress, and we call this technique the **learning heavy Fourier coefficients**.

3.1 Algorithm Model

Given a function f and a threshold θ



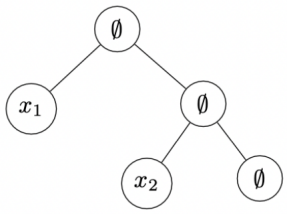
- Output all coefficient S such that $\left|\hat{f}(S)\right| \geq \theta$ (all interesting).
- Only output S such that $\left|\hat{f}(S)\right| \geq \frac{\theta}{2}$ (no junk).

We set the threshold to be θ, so this query will return interesting coefficients. However, this won't guarantee that we won't miss some of them, so we also output S with Fourier above the "not-too-low" threshold to avoid a sampling error.

3.2 Main Idea

We exhaustively search for all big S's with a good pruning. We represent the pruning via a tree. If x_k is in S, the path goes to the left at level k; otherwise, it goes to the right at level k. Note that the Goldreich-Levin and Kushilevitz-Mansour learning algorithms have similar trees but are different in the search method.

Level	Variable
1	x_1
2	x_2
$\vdots$	$\vdots$
n	x_n



This search aims to go down a subtree having a large sum of "weight", which usually means the sum of the square of Fourier coefficients. Then, the algorithm will output merely leaves reached the bottom (ie. reaching level n).

Definition 9 *For a fixed current level $0 \leq k \leq n$ and a current node of search $S_1 \subseteq [k]$, we define a function $f_{k,S_1} : \{\pm 1\}^{n-k} \to \mathbb{R}$ such that*

$$f_{k,S_1}(x) = \sum_{T_2 \subseteq \{k+1,\ldots,n\}} \hat{f}(S_1 \cup T_2)\chi_{T_2}(x).$$

We used χ_{T_2} rather than $\chi_{S_1 \cup T_2}$ because $\chi_{S_1 \cup T_2} = \chi_{S_1} \cdot \chi_{T_2}$ and χ_{S_1} is constant as we fixed the current node. In fact, $\chi_{S_1} = 1$ since we selected all first k variables (ie. $x_1 = \cdots = x_k = 1$). We can examine some specific k for example:

Case $k = 0$ (root)

$$f_{0,\emptyset}(x) = \sum_{T_2 \subseteq [n]} \hat{f}(S_1 \cup T_2)\chi_{T_2}(x) = \sum_{T_2 \subseteq [n]} \hat{f}(T_2)\chi_{T_2}(x) = f(x).$$

Case $k = n$ (leaf)

$$f_{n,S_1}(x) = \sum_{T_2 \subseteq \emptyset} \hat{f}(S_1 \cup T_2)\chi_{T_2}(x) = \hat{f}(S_1)$$

since $\hat{f}(S_1 \cup T_2) = \hat{f}(S_1)$ and $\chi_\emptyset(x) = 1$.

3.3 Plan

The pruning should only go down paths with $\mathrm{E}_x\left[f_{k,S_1}(x)^2\right] \geq \theta^2$. Some aspects we have to keep in mind about this algorithm:

1. Computation: How to compute $f_{k,S_1}(x)$?
2. Correctness: Will it reach to right leaves, all heavy leaves without too much junk?
3. Runtime: How many paths do we take?

We will first focus on how many paths to ensure that we won't examine too many of them.

3.4 Explore Paths

Next, we will prove the following lemma to show that we won't explore too many nodes not only just at the bottom but at any level k.

Lemma 10 (Not too many paths) *For a boolean function $f : \{\pm 1\}^n \to \{\pm 1\}$,*

(i) At most $\frac{1}{\theta^2}$ of S's such that $\left|\hat{f}(S)\right| \geq \theta$

(ii) For any integer $k \in [0, n]$, there are at most $\frac{1}{\theta^2}$ functions f_{k,S_1} with $E_x\left[f_{k,S_1}(x)^2\right] \geq \theta^2$

Then, we explore at most $O\left(\frac{1}{\theta^2}\right)$ nodes

Proof

(i) By Parserval's identity on a boolean function, we get $1 = \sum_S \hat{f}(S)^2$. If there are more than $\frac{1}{\theta^2}$ of S's such that $\left|\hat{f}(S)\right| \geq \theta$, then $\sum_S \hat{f}(S)^2 > \frac{1}{\theta^2} \cdot \theta^2 = 1$, which is contradiction!

(ii) First, we show that

$$\begin{aligned}
\mathrm{E}_x\left[f_{k,S_1}(x)^2\right] &= \mathrm{E}_x\left[\left(\sum_{T_2} \hat{f}(S_1 \cup T_2)\chi_{T_2}(x)\right)^2\right] \\
&= \mathrm{E}_x\left[\sum_{T_2,T_2'} \hat{f}(S_1 \cup T_2)\hat{f}(S_1 \cup T_2')\chi_{T_2}(x)\chi_{T_2'}(x)\right] \\
&= \sum_{T_2,T_2'} \hat{f}(S_1 \cup T_2)\hat{f}(S_1 \cup T_2')\mathrm{E}_x\left[\chi_{T_2}(x)\chi_{T_2'}(x)\right] \\
&= \sum_{T_2} \hat{f}(S_1 \cup T_2)^2.
\end{aligned}$$

The last line of equation follows from

$$\mathrm{E}_x\left[\chi_{T_2}(x)\chi_{T_2'}(x)\right] = \begin{cases} 1, & \text{if } T_2 = T_2' \\ 0, & \text{otherwise} \end{cases}$$

because $\chi_{T_2}(x)$ and $\chi_{T_2'}(x)$ are different signs half of the time if $T_2 \neq T_2'$. Then, using Parserval's identity, we have

$$1 = \sum_{S \subseteq [n]} \hat{f}(S)^2 = \sum_{S_1 \subseteq [k]} \sum_{T_2 \subseteq \{k+1,\ldots,n\}} \hat{f}(S_1 \cup T_2)^2 = \sum_{S_1 \subseteq [k]} \mathrm{E}_x\left[f_{k,S_1}(x)^2\right]$$

Thus, we have at most $\frac{1}{\theta^2}$ of S_1's such that $\mathrm{E}_x\left[f_{k,S_1}(x)^2\right] > \theta^2$

■