0368.4163 Randomness and Computation June 3, 2009

Lecture 11

Lecturer: Ronitt Rubinfeld *Scribe: Anat Ganor, Roy Kasher, O. Paneth*

1 Introduction

In this lecture we show an algorithm for learning Parity with noise. If f is linear we can find it easily by solving a linear system of equations. For an arbitrary f, our goal is to find all linear functions that are close to f.

Under the general PAC model this problem is conjectured to be hard, refered to as *hardness of parity with noise*, *hardness of decoding linear codes*, *maximum likelihood decoding of linear codes* or *finding largest fourier coefficient*. For certain relaxtions of the problem we know subexponential algorithms exist, however, the general case is believed to be NP-hard.

We can consider two noise models: Adverserial, where a linear function is changed adverserially in up to $\frac{1}{2} - \epsilon$ fraction of its inputs, or noisy, where the corruption occurs randomly.

Since the problem is infeasible in the PAC model, we will allow queries to f instead of random sampling. This makes it feasible even with adverserial noise. This problem was first studied by Goldreich and Levin in the context of cryptographic pseudorandom generators, and has found applications in error correcting coes (list decoding of Hadamard codes) and learning theory (by Kushilevitz and Mansour).

The goal We want to output all $S \subseteq [n]$ s.t. $\hat{f}(S) \geq \theta$ (i.e. $\Pr_x [\chi_S(x) = f(x)] \geq \frac{1+\theta}{2}$) and not output any S for which $\hat{f(S)} \leq \frac{\theta}{2}$, in runtime poly$(n, \frac{1}{\theta})$

Notation

- $e_i = (1, ..., 1, -1, 1, ..., 1)$ where -1 is in the i^{th} location.
- Bitwise product: $xy = (x_1 y_1, x_2 y_2, ..., x_n y_n)$

2 Learning Parity with Noise

2.1 Warmup 1

f is linear, i.e. $\exists S$ s.t. $f(x) = \chi_S(x)$

The algorithm
$\forall i \in [n]$ put i in S if $1 = f(\overrightarrow{1}) \neq f(e_i)$

2.2 Warmup 2

Arbitrary f, using poly queries and unbounded time

The algorithm
Iterate all S and test if f and χ_S agree "enough" (by sampling)

Note the samples must be "recycled" for all S's. The analysis is simple using the Chernoff and union bounds as we've seen in the last lecture.

2.3 Warmup 3

Exists S s.t. f agrees with χ_S "almost everywhere", on $\geq 1 - \frac{1}{4n}$ fraction of inputs (Note every two linear functions disagree on half of the inputs, so f is determined uniquely by χ_S)

The algorithm

- Choose $r \leftarrow_R \{\pm 1\}^n$
- $\forall i \in [n]$ put i in S if $f(r) \neq f(r \cdot e_i)$

Why does it work?

$$\begin{aligned}
\Pr[i \text{ is decided incorrectly}] &\leq \Pr[f(r) \neq \chi_S(r) \text{ or } f(r \cdot e_i) \neq \chi_S(r \cdot e_i)] \\
&\leq \Pr[f(r) \neq \chi_S(r)] + \Pr[f(r \cdot e_i) \neq \chi_S(r \cdot e_i)] \text{ (union bound)} \\
&\leq \frac{1}{4n} + \frac{1}{4n} = \frac{1}{2n} (r \text{ and } r \cdot e_i \text{ are uniformly distributed})
\end{aligned}$$

$\Rightarrow \Pr[\text{any } i \text{ is decided incorrectly}] \leq n \cdot \frac{1}{2n} = \frac{1}{2}$ (union bound)

2.4 Warmup 4

Exists S s.t. f agrees with χ_S on $\geq \frac{3}{4} + \epsilon$ fraction of the inputs, for a constant ϵ. Note the previous algorithm doesn't work. We will use standard amplification techniques.

The algorithm

- Choose $r_1, r_2, ..., r_t \leftarrow_R \{\pm 1\}^n$ where $t = \Theta(\frac{\log(n)}{\epsilon^2})$
- $\forall i \in [n]$ put i in S if for the majority of r_j's $f(r_j) \neq f(r_j \cdot e_i)$

Why does it work?
$\Pr[\text{wrong answer for } i \text{ on } r_j] \leq \frac{1}{2} - 2\epsilon$
$\Rightarrow \Pr[\text{most } r_j\text{'s wrong for } i] \leq \frac{1}{2n}$ (by Chernoff and the choice of t)
$\Rightarrow \Pr[\text{any } i \text{ is decided incorrectly}] \leq \frac{1}{2}$ (union bound)

2.5 Warmup 5

Exists S s.t. f agrees with χ_S on $\geq \frac{1}{2} + \epsilon$ fraction of the inputs, for a constant ϵ. Note we cannot win with the union bound as before, and we must decide on i using only one query. To do this, we can just assume we are given the correct values of $\sigma_j = \chi_S(r_j)$. The problem now becomes that of testing $\sigma_j \neq f(r_j \cdot e_i)$ using the previous techniques. Since we can eventually verify (by sampling) that our output function is indeed close to f, we can simply enumerate all possible values of the σ_j's.

The algorithm

- Choose $r_1, r_2, ..., r_t \leftarrow_R \{\pm 1\}^n$ where $t = \Theta(\frac{\log(n)}{\epsilon^2})$
- *Generating candidates*
  For all possible settings of $\sigma_1\sigma_2, ...\sigma_t$ (guesses of $\chi_S(r_j)$'s)
  $\forall i \in [n]$ put i in $S_{\sigma_1\sigma_2...\sigma_t}$ if for the majority of r_j's $\sigma_j \neq f(r_j \cdot e_i)$

- *Verify the candidates before outputting*
  Test $\chi_{S_{\sigma_1\sigma_2\ldots\sigma_t}}$ agrees with f on $\geq \frac{1}{2} + \frac{3}{4}\epsilon$ fraction of inputs by sampling

Why does it work?
For each S that should be output there are $\sigma_1, \sigma_2, ..., \sigma_t$ s.t. $\forall j\, \sigma_j = \chi_S(r_j)$.
The test puts i in S if $\sigma_j \cdot f(r_j \cdot e_i) = -1$. The test succeeds when $f(r_j \cdot e_i) \cdot \sigma_j = \chi_S(e_i)$. Note that if $f(r_j \cdot e_i) = \chi_S(r_j \cdot e_i)$ then $f(r_j \cdot e_i) \cdot \chi_S(r_j) = \chi_S(r_j \cdot e_i) \cdot \chi_S(r_j) = \chi_S(e_i)$ and the test succeeds.

$$\begin{aligned}\Pr[\text{right answer for } r_j \text{ on } i] &= \Pr[f(r_j \cdot e_i) \cdot \sigma_j = \chi_S(e_i)] \\ &\geq \Pr[f(r_j \cdot e_i) = \chi_S(r_j \cdot e_i)] \\ &\geq \frac{1}{2} + \epsilon\end{aligned}$$

$\Rightarrow \Pr[\text{most } r_j\text{'s wrong for } i] \leq \frac{1}{2n}$ (by Chernoff and the choice of t)
$\Rightarrow \Pr[\text{any } i \text{ is decided incorrectly}] \leq \frac{1}{2}$ (union bound)

This analysis shows that if the σ_j's are consistent with χ_S and χ_S is close to f, then it will be output w.p. at least half, which can easily be amplified. Since we enumerate all the possible σ_j's, it remains to show that no far functions will be output. This is guaranteed from the verification stage that ensures functions that are $(\frac{1}{2} + \frac{\epsilon}{2})$-far are output with low probability.

2.6 Finally – The Real Problem

Exists S s.t. f agrees with χ_S on $\geq \frac{1}{2} + \epsilon$ fraction of the inputs, for *any* ϵ. The last algorithm we've seen requires ϵ to be constant for Chernoff to work for small t's. Larger t's are not advised as we enumerate 2^t possible σ's. Since we are dealing with linear functions, we can guess a much smaller set and compute the function values in all possible linear combinations.

The algorithm

- Choose $u_1, u_2, ..., u_k \leftarrow_R \{\pm 1\}^n$ where $k = \log(t+1)$, $t = \Theta(\frac{n}{\epsilon^2}$ (number of r_j's)
- *Generating lots of examples ($\Theta(\frac{n}{\epsilon^2})$)*
  For all possible settings of $\sigma_1, \sigma_2, ..., \sigma_k$ (guesses of $\chi_S(u_j)$'s)
  For every $w \subseteq \{1, ..., k\}$, $w \neq \emptyset$
  Set $r_w \leftarrow \oplus_{j \in w} u_j$ and $p_w \leftarrow \prod_{j \in w} \sigma_j$
- *Generating candidates*
  For all possible settings of $\sigma_1, \sigma_2, ..., \sigma_k$ (guesses of $\chi_S(u_j)$'s)
  $\forall i \in [n]$ put i in $S_{\sigma_1\sigma_2\ldots\sigma_k}$ if for the majority of r_j's $p_w \neq f(r_w \cdot e_i)$
- *Verify the candidates before outputting*
  Test $\chi_{S_{\sigma_1\sigma_2\ldots\sigma_k}}$ agrees with f on $\geq \frac{1}{2} + \frac{3}{4}\epsilon$ fraction of inputs by sampling

Why does it work?
As before, for a "given" S the test succeeds for r_w when $f(r_w \cdot e_i) \cdot p_w = \chi_S(e_i)$, and this happens w.p. $\geq \frac{1}{2} + \epsilon$. Let X_w be an indicator for this event. Unfortunately, unlike in the last warmup, we can no longer claim independence of the indicators and apply the Chernoff bound. However, as in the constructions in lecture 3, the r_w's are *pairwise independent*.
We have: $\mathrm{E}[X_w] \geq \frac{1}{2} + \epsilon$, $\mathrm{VAR}[X_w] = \mathrm{E}[X_w^2] - \mathrm{E}[X_w] \leq \frac{1}{4} - \epsilon^2$. By Chebychev,

$$\begin{aligned} Pr\left[\text{most } r_j\text{'s wrong for } i\right] &= \Pr[\sum X_w < \frac{t}{2}] \\ &\leq \Pr[\left|\frac{1}{n}\sum X_w - \mathrm{E}[X_w]\right| > \epsilon] \\ &\leq \frac{\mathrm{VAR}[X_w]}{\epsilon^2 t} \leq \frac{1}{2n} \end{aligned}$$

$\Rightarrow \Pr\left[\text{any } i \text{ is decided incorrectly}\right] \leq \frac{1}{2}$ (union bound)

Theorem 1 *There is an algorithm which outputs all S s.t. $|\hat{f}(S)| \geq \theta$ and doesn't output any S s.t. $|\hat{f}(S)| \leq \frac{\theta}{2}$ w.p. $\geq 1-\delta$ with query and time complexity* $\mathrm{poly}(n, \frac{1}{\theta}, \log(\frac{1}{\delta}))$

Corollary 2 *The number of sets S s.t. $|\hat{f}(S)| \geq \theta$ is at most $O(\frac{n}{\theta^2})$*

Theorem 3 *Given $\mathcal{S} \subseteq 2^{[n]}$ (a collection of subsets) s.t. $\sum_{S\in\mathcal{S}} \hat{f}(S)^2 \geq 1-\epsilon$ and* $\mathrm{poly}(n, |\mathcal{S}|, \frac{1}{\tau}, \log(\frac{1}{\delta}))$ *uniformly chosen examples of f, there is an algorithm which outputs $g(x) : \{\pm 1\}^n \to \mathbb{R}$ s.t. $g(x) = \sum_{S\in\mathcal{S}} C_S \chi_S(x)$ and $\Pr\left[sign(g(x)) \neq f(x)\right] \leq \epsilon + \tau$ w.p. $\geq 1-\delta$*

Proof Idea Use previous theorem and the ideas from the low degree algorithm of last lecture. ■

3 Learning Functions with Small L_1 Norm

Definition 1 *The L_1 norm of f is $L_1(f) = \sum_S |\hat{f}(S)|$*

Examples:

- $L_1(\chi_S) = 1$
- $L_1((x_1 \wedge x_2) \oplus (x_3 \wedge x_4) \oplus ...) = 2^{\frac{n}{2}}$

Claim 4 *Given ϵ, $\mathcal{S}_\epsilon = \left\{S \subseteq [n] : |\hat{f}(S)| \geq \frac{\epsilon}{L_1(f)}\right\}$ then:*

1. $|\mathcal{S}_\epsilon| \leq \frac{L_1(f)^2}{\epsilon}$
2. $\sum_{S\in\mathcal{S}_\epsilon} \hat{f}(S)^2 \geq 1-\epsilon$

Proof

1. $L_1(f) \geq \sum_{S\in\mathcal{S}_\epsilon} |\hat{f}(S)| \geq \frac{\epsilon}{L_1(f)}|\mathcal{S}_\epsilon|$
2. $\sum_{S\notin\mathcal{S}_\epsilon} \hat{f}(S)^2 \leq \max_{S\notin\mathcal{S}_\epsilon} |\hat{f}(S)| \cdot \sum_{S\notin\mathcal{S}_\epsilon} |\hat{f}(S)| \leq \frac{\epsilon}{L_1(f)} L_1(f) = \epsilon$. By Boolean Parseval, $\sum_S \hat{f}(S)^2 = 1$ and so $\sum_{S\in\mathcal{S}_\epsilon} \hat{f}(S)^2 \geq 1-\epsilon$

■

Theorem 5 *We can learn any Boolean function to ϵ-accuracy with membership queries in time* $\mathrm{poly}(n, L_1(f), \frac{1}{\epsilon})$

Proof Use theorems 1 and 3 with $\theta = \epsilon L_1(f)$. If $L_1(f)$ is unknown, try values $2, 4, 8, \ldots$ for $L_1(f)$ until you find a good hypothesis (Can always test it by random sampling). ■

3.1 Learning Decision Trees

Theorem 6 *If f has decision tree with $\leq t$ leaves then $L_1(f) \leq t$*

Proof Fix leaf l. Define $g_l = I[x \text{ reaches } l]$, let $x_{i_1}, \ldots, x_{i_k}$ be the path to l, and m the number of left turns taken. Recall $g_l(x) = \left(\frac{1 \pm x_{i_1}}{2}\right) \cdot \left(\frac{1 \pm x_{i_2}}{2}\right) \cdot \ldots \cdot \left(\frac{1 \pm x_{i_k}}{2}\right) = \frac{1}{2^k} \sum_{S \in [k]} (-1)^m \chi_S(x)$ and so $L_1(g_l) = \sum_{S \in [k]} |\hat{(g)}_l(S)| = 2^k \cdot \frac{1}{2^k} = 1$
Since $f(x) = \sum_l g_l(x)(\text{output of leaf } l)$ we have $\hat{f}(x) = \sum_l \hat{g}_l(x)(\text{output of leaf } l)$

$\Rightarrow L_1(f) = \sum_S |\hat{f}(S)| = \sum_S \sum_l |\hat{g}_l(x) \cdot (\pm 1)| = \sum_l \sum_S |\hat{g}_l(x)| = \sum_l L_1(g_l) = \sum_l 1 = t$ ■