6.895 Randomness and Computation March 5, 2008

Lecture 9

Lecturer: Ronitt Rubinfeld *Scribe: Andrew Correa*

1 Last Time

Today we look at the problem of learning Fourier coefficients with queries. The goal is to output all S s.t. $\hat{f}(S) \geq \theta$ and all S that are output should have $\hat{f}(S) \geq \frac{\theta}{2}$.

We explore the following full binary tree. The 2^k nodes on the k-th level, $0 \leq k \leq k$, correspond to all subsets of $[k]$. A node $S \subseteq [k]$ on the k-th level has two children that correspond to S and $S \cup \{k+1\}$. The leaves of the tree correspond to all subsets of $[n]$, and therefore to all Fourier coefficients.

For a node $S_1 \subseteq [k]$ on the k-the level, we define

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

Our plan is to go down subtrees of value $\mathbb{E}_x[f^2_{k,S_1}(x)] \geq \frac{\delta^2}{2}$.

2 Learning Fourier Coefficients with Queries

So far we looked at the case where we had no choice of which x to use. We had to make the best of the pairs $(x, f(x))$ we were given. This time we look at the case when we are allowed to choose which x to query, so we can take advantage of that. The basic idea is that we will be using a exhaustive search of a (hopefully very) pruned binary tree. How do we prune it? We will be using an "amazing" oracle.

Remember Parseval's: $1 = \mathbb{E}_x[f^2(x)] = \sum_S \hat{f}^2(S)$.

Claim 1 $\forall k, S_1 \subseteq [k]$

$$\mathbb{E}_x[f^2_{k,S_1}(x)] = \sum_{T_2} \hat{f}^2(S_1 \cup T_2)$$

Proof This follows directly from Parseval's as $\hat{f}^2(S_1 \cup T_2)$ is a Fourier coefficient of $f^2_{k,S_1}(x)$. ■

Now we must prove that we are traversing an appropriate number (i.e. not too many and not too few) nodes of the tree. In other words, we must prove both that we are pruning bad x whose $\mathbb{E}[\hat{f}^2_{k,S_1}(x)] < \frac{\theta^2}{2}$ and at the same time traversing into nodes that represent the x whose $\mathbb{E}[\hat{f}^2_{k,S_1}(x)] \geq \frac{\theta^2}{2}$.

Fact 2 (Not too Few) *For all subsets S_1, if there exists a $\tilde{T}_2$ such that $|\hat{f}(S_1 \cup \tilde{T}_2)| > \theta$, then:*

$$\mathbb{E}_x[f^2_{k,S_1}(x)] = \sum_{T_2} \hat{f}^2(S_1 \cup T_2) \geq \hat{f}^2(S_1 \cup \tilde{T}_2) \geq \theta^2$$

Thus we know that we will not visit too few nodes in the tree.

... but what if we visit too many ...

Lemma 3 (Not too Many) *For all $\theta > 0$, we have:*

1. *Less than $\frac{1}{\theta^2}$ S's satisfy $|\hat{f}(S)| \geq \theta^2$.*
2. *For all $0 \leq k \leq n$, less than $\frac{1}{\theta^2}$ functions f_{k,S_1} have $\mathbb{E}_x[f^2_{k,S_1}(x)] \geq \theta^2$.*

Notice two things. First, notice that Part 1 bounds the number of returned nodes S and Part 2 bounds the running time, so both the amount of returned data *and* the running size are bounded from above. Second, notice that while the the actual values will differ slightly from those above, it will be by only a constant factor.

Proof

1. Assume that Part 1 (above) is false. Then:

$$1 = \sum_S \hat{f}^2(S) > \left(\frac{1}{\theta^2}\right) \cdot \theta^2 = 1$$

   And thus $1 > 1$ which is a contradiction.

2. Assume that Part 2 (above) is false. Then given k:

$$\begin{aligned} 1 &= \sum_S \hat{f}^2(S) \\ &= \sum_{S_1 \subseteq [k]} \sum_{T_2 \subseteq \{k+1,\ldots,n\}} \hat{f}^2(S_1 \cup T_2) \\ &= \sum_{S_1 \subseteq [k]} \mathbb{E}_x[f^2_{k,S_1}(x)] \\ &> \left(\frac{1}{\theta^2}\right) \cdot \theta^2 \;=\; 1 \end{aligned}$$

   And thus $1 > 1$, which is (still) a contradiction.

■

Now that we know that the algorithm neither leaves anything out nor selects too much, how can we speed it up? How can we quickly *estimate* $f_{k,S_1}(x)$?

Lemma 4 ($f_{k,S_1}(x)$ Estimation)

$$f_{k,S_1}(x) = \mathbb{E}_{y \in \{\pm 1\}^k}[f(yx)\chi_{S_1}(y)]$$

Note that this could be estimated by sampling. In general,Thus, picking random y's and outputting the average value gives γ-additive approximation by Chernoff Bounds in $\mathrm{O}(\frac{1}{\gamma^2}\log\frac{1}{\delta})$ (where δ is the security parameter). Then we can use these estimates to estimate $\mathbb{E}_x[f_{k,S_1}(x)]$ also by Chernoff Bounds.
Proof First notice some (maybe obvious) things:

- $f(yx) = \sum_T \hat{f}(T)\chi_T(yx)$
- For $T = T_1 \cup T_2$, where $T_1 \subseteq [k]$, and $T_2 \subseteq \{k+1, \ldots, n\}$,

$$\chi_T(yx) = \chi_{T_1}(y)\chi_{T_2}(x).$$

Now the proof:

$$\begin{aligned}
\mathbb{E}_y\left[f(yx)\chi_S(y)\right] &= \mathbb{E}_y\left[\underbrace{\sum_{T_1}\sum_{T_2}\hat{f}(T_1\cup T_2)\chi_{T_1}(y)\chi_{T_2}(x)}_{f(yx)}\,\chi_{S_1}(y)\right] \\
&= \sum_{T_1}\sum_{T_2}\hat{f}(T_1\cup T_2)\chi_{T_2}(x)\underbrace{\mathbb{E}_y\left[\chi_{T_1}(y)\chi_{S_1}(y)\right]}_{0 \text{ unless } T_1=S_1} \\
&= \sum_{T_2}\hat{f}(S_1\cup T_2)\chi_{T_2}(x) \\
&= f_{k,S_1}(x)
\end{aligned}$$

■

So the algorithm then becomes:

1. If $k=n$, output S_1.
2. Else:
   (a) If estimate of $\mathbb{E}[f^2_{k+1,S_1\cup\{k+1\}}(x)] \geq \frac{\theta^2}{2}$, recurse on $(k+1, S_1\cup\{k+1\})$.
   (b) If estimate of $\mathbb{E}[f^2_{k+1,S_1}(x)] \geq \frac{\theta^2}{2}$, recurse on $(k+1, S_1)$.

Theorem 5 *For all $\theta > 0$, this algorithm outputs $\mathcal{S} = \{S_1, \ldots, S_l\}$ such that $l = \theta\left(\frac{1}{\theta^2}\right)$ such that with probability greater than $1-\delta$:*

1. $\forall S_i \in \mathcal{S}, \quad |\hat{f}(S)| \geq \frac{\theta}{2}$,

2. $\forall S_i \notin \mathcal{S}, \quad |\hat{f}(S)| < \theta$,

with query complexity $\mathrm{poly}(n, \frac{1}{\theta}, \log\frac{1}{\delta})$.

Proof

1. If $\hat{f}(S) < \frac{\theta}{2}$, then $\hat{f}^2(S) \leq \frac{\theta^2}{4}$
2. If $\hat{f}(S) > \theta$, then all ancestors of S will continue the recursion.

The total number of nodes explored will be less than $\mathrm{O}(\frac{n}{\theta^2})$. ■

After we learn a small set of significant Fourier coefficients, we can use the following theorem to compute a function close to f.

Theorem 6 *There exists an algorithm, which given $\mathcal{S} \subseteq 2^{[n]}$ such that $\sum_{S\in\mathcal{S}}\hat{f}^2(S) \geq 1-\varepsilon$ and examples, with probability $1-\delta$ outputs $g : \{\pm1\}^n \mapsto \mathbb{R}$ such that:*

- $g(x) = \sum_{S\in\mathcal{S}} C_S\chi_S(x)$
- $\Pr[f(x) \neq \mathrm{sign}(g(x))] \leq \varepsilon + \tau$
- *The number of examples/queries is* $\mathrm{poly}(n, |S|, \frac{1}{\tau}, \log\frac{1}{\delta})$.

Proof The same method as proving the low degree algorithm ■

In general: *Good* functions for our approach are functions like the parity functions χ_S (only one non-zero Fourier coefficients). *Problem* functions are, for instance, functions of the form:

$$P_2 = (x_1 \wedge x_2) \oplus \ldots \oplus (x_{n-1} \wedge x_n) = \sum_S \pm 2^{-\frac{n}{2}} \cdot \chi_S$$

since they have many small Fourier coefficients.

Definition 7 *For a function $f : \{\pm 1\}^n \mapsto \mathbb{R}$ the L_1 norm of f is $\sum_S |\hat{f}(S)|$.*

Notice that for the above example of good and bad functions, the L_1 norm of a good function is 1 and the L_1 norm of a bad function is $2^{n/2}$.

Claim 8 *Let $f : \{\pm 1\}^n \mapsto \{\pm 1\}$ be a function. Given ε, let $\mathcal{S}_\varepsilon = \left\{ S \subseteq [n] \,\middle|\, |\hat{f}(S)| \geq \frac{\varepsilon}{L_1(f)} \right\}$. It holds:*

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

Proof

1. $\sum_{S \notin \mathcal{S}} \hat{f}^2(S) \leq \frac{\varepsilon}{L_1(f)} \sum_S |\hat{f}(S)| \leq \varepsilon$
2. $|\mathcal{S}_\varepsilon| \frac{\varepsilon}{L_1(f)} \leq \sum_{S \in \mathcal{S}_\varepsilon} |\hat{f}(S)| \leq \sum_{S \in [n]} |\hat{f}(S)| = L_1(f)$

■

Theorem 9 *We can learn any Boolean function f to ε-accuracy with queries in time* $\mathrm{poly}(n, L_1(f), \frac{1}{\varepsilon})$.

Sketch of Proof Suppose first that we know $L_1(f)$. We first learn all coefficients in S_ε (plus perhaps a few other coefficients). We are interested in coefficients $\hat{f}(S) \geq \frac{\varepsilon}{L_1(f)}$, and we run the algorithm that uses queries to find a set of size $O(|S_\varepsilon|)$ that contains all of them. Then, we run the algorithm of Theorem 6 to compute f's approximation.

But we are not given $L_1(f)$, so what should we do? We try setting $L_1(f)$ to 1, 2, 4, 8, ... Each time we compare the g that we obtain to f on random samples to test if we are done. ■