6.895 Randomness and Computation February 27, 2006

Lecture 6

Lecturer: Ronitt Rubinfeld *Scribe: Benjamin Snyder*

Our general goal over the course of the next few lectures will be the following: given some $f : \{\pm 1\} \to \{\pm 1\}$, output $h : \{\pm 1\} \to \Re$ s.t. $\Pr_x[f(x) \neq sgn(h(x))] \leq \varepsilon$.

In particular, we will be working in the framework known as "Learning with Queries under the Uniform distribution."

1. We are allowed queries of the form, "what is $f(x)$?" (some learning frameworks allow more general questions). In fact, we will see that in our first example we will only require query access for randomly chosen values of x, instead of for arbitrary values of our own choosing.
2. Error is computed under a uniform distribution. i.e. no "special inputs"
3. We won't know which inputs are likely to produce errors – we just have a bound on the total error.
4. If f is random (in the sense that a bit is chosen at random to be the output for a particular input), then there is no way to find such an h in less than exponential time, as we would have to sample each possible input. Instead we will look at learning f when it comes from certain special families of functions such as (i) constant depth polysize circuits, and (ii) the set of "small" decision trees.
5. Finally, it should be noted that the resulting function h need not be from the same special family as f.

Our algorithms will find a function that approximates f by approximating certain subsets of the Fourier coefficients. In the case of small depth circuits, we will approximate all Fourier coefficients corresponding to small sets S, and in the case of decision trees, we will approximate all Fourier coefficients that have large enough absolute value. Thus we will have two potential sources of error – the first is that we are completely ignoring most Fourier coefficients and the second is that even for those that we are considering, we only get an approximation.

Note however, that we can approximate any Fourier coefficient with arbitrary precision using sampling in the following way: for any given set $S \subseteq [n]$, with $O(\frac{k}{\delta^2})$ samples we can approximate $\hat{f}(S) = 1 - 2\Pr_x[f(x) \neq \chi_S(x)]$ (an equality proved in a previous lecture) with some value $\tilde{f}$, where $\Pr[|\tilde{f}(S) - \hat{f}(S)| > \delta] \leq e^{-k}$. This is done by sampling values of x and estimating $\Pr_x[f(x) \neq \chi_S(x)]$. The proof proceeds with a straighforward application of Chernoff bounds.

Of course, we will still need to analyze the effect these approximations have on the overall error of the outputted function h for each of the individual algorithms.

1 Learning Circuits

First we take up the issue of learning f when it is in the family of constant depth polysize circuits.

input: $X_1, X_2, ..., X_n \in \{0,1\}^n$
gates: $\vee, \wedge, \neg$

Our $\vee$ and $\wedge$ will have unbounded fan-in (i.e. they can take any subset of the variables as inputs).

Also, by DeMorgan's law we can always push negation into conjunctive and disjunctive expressions (e.g. $\neg(X_1 \wedge X_2) = \neg X_1 \vee \neg X_2$). We will assume that that this has been and that the negation gates

will always occur (if at all) at the first level of the circuit, and we will therefore not include negation gates in its size.

Now we will switch back to the bit notation of $\{\pm 1\}$ and assume that $\wedge$, $\vee$ and $\neg$ have been mapped to operations with equivalent truth-tables (i.e. where $+1$ takes the place of 0, and -1 takes the place of 1).

Definition 1 *Let $\mathcal{F}(s,d)$ be the family of functions computable via circuits of size $\leq s$ and depth $\leq d$*

For example, $\mathcal{F}(2^n, 2)$ is the set of all n-bit functions (since the entire truth-table can be directly encoded using 2^n gates).

Question: does $\mathcal{F}(\text{poly}(n), d)$ (known as AC^0) contain the parity function?

This problem was answered in the negative by a series of works due to several authors. In fact, what this line of work was able to show was the following:

Theorem 2 (Hastad, Linial Mansour Nisan) *For any function f in $\mathcal{F}(s,d)$, and for $t = 28 \times (14 \times \log(\frac{2s}{\alpha}))^{d-1}$, we have $\sum_{|S|>t} \hat{f}^2(S) \leq \alpha$.*

To make sense of the value of t as we will use it, consider the case where $s = \text{poly}(n)$, d is a constant, and α is an error constant. Then we have $t = O(\log^d(\frac{n}{\alpha}))$, and we can understand the theorem intuitively as saying that Fourier coefficients of large sets S (i.e. of linear functions with large multipliers) can't have too much energy.

Now note that the parity function places all its weight on the single Fourier coefficient $\hat{f}([n])$ and therefore by Parseval's identity, for any $t < n, \sum_{|S|>t} \hat{f}^2(S) = 1$. Then, by Theorem 2, AC^0 does *not* contain parity.

This is the bad news. However, some good news can also be found in Theorem 2 as the following results shows.

Theorem 3 (LMN) *For any function $f \in \mathcal{F}(s,d), \exists$ an algorithm that in time only $n^{O(\log(\frac{s}{\varepsilon}))^{d-1}}$ returns a function g s.t. $\Pr_x[f(x) \neq sgn(g(x))] \leq \varepsilon$.*

Corollary 4 *For any function $f \in \mathcal{F}(\text{poly}(n), d), \exists$ an algorithm that in time only $n^{O(\log(\frac{n}{\varepsilon}))^{d-1}}$ returns a function g s.t. $\Pr_x[f(x) \neq sgn(g(x))] \leq \varepsilon$.*

Algorithm

Input: function f, computable by some circuit of size $s = poly(n)$, constant depth d.
Let $\alpha = \varepsilon/2$, $t = 28 \times (14 \times \log(\frac{2s}{\alpha}))^{d-1}$, and $\delta = \sqrt{\frac{\varepsilon}{2n^t}}$
Use sampling to compute $\tilde{f}(S)$, for S s.t. $|S| < t$, to within error δ of $\hat{f}(S)$.
Return $g(x) = \sum_{|S|<t} \tilde{f}(S)\chi_S(x)$.

We need $O(\frac{k}{\delta^2}) = O(\frac{n^t}{\varepsilon})$ samples to estimate each $\tilde{f}(S)$ within error δ of $\hat{f}(S)$ with confidence $1 - e^{-k}$.

Now,

$$\begin{aligned} \Pr_x[g(x) \neq f(x)] &\leq \text{Exp}[(f(x) - g(x))^2] \\ &= \sum_S (\hat{f}(S) - \hat{g}(S))^2 \end{aligned}$$

$$
\begin{aligned}
&= \sum_{|S|<t} (\hat{f}(S) - \tilde{f}(S))^2 + \sum_{|S|>t} \hat{f}(S)^2 \\
&\leq \sum_{|S|<t} \delta^2 + \alpha \\
&= \sum_{i=1}^{t} \binom{n}{i} \delta^2 + \alpha \\
&\leq n^t \delta^2 + \alpha \\
&= \varepsilon/2 + \varepsilon/2 \\
&= \varepsilon
\end{aligned}
$$

2 Learning Decision Trees

Recall from last time the Goldreich-Levin (GL) algorithm, whose existence can be stated as the following theorem:

Theorem 5 *$\exists$ an algorithm that given parameters Θ and oracle access to $f : \{\pm 1\}^n \to \{\pm 1\}$, outputs in time* poly$(n, \frac{1}{\Theta})$ *a list L of subsets of $[n]$ that with high probability finds all $S \subseteq [n]$ with $|\hat{f}(S)| \geq \Theta$ (and contains no S s.t. $|\hat{f}(S)| \leq \frac{\Theta}{2}$)*

Definition 6 *An* m-node decision tree *is any tree of size m whose internal nodes are labeled by values $i \in [n]$ and whose leaf nodes are labeled by values $j \in \{\pm 1\}$.*

Definition 7 *A function $f : \{\pm 1\}^n \to \{\pm 1\}$ is* computable by an m-node decision tree *if for $\forall x \in \{\pm 1\}^n, \exists$ an m-node decision tree s.t. the leaf reached from the traversal of x down the tree has value $f(x)$. A traversal is governed by the rule that at a node labeled with i, if $x_i = -1$, the traversal continues down the left child, and if $x_i = +1$, the traversal continues down the right child.*

Claim 8 *For any function f computable by an m-node decision tree, the L_1 norm of f (defined to be $\sum_Z |\hat{f}(Z)|$) is small (i.e. $\leq m$).*

Claim 9 *When $L_1(f)$ is small, the large Fourier coefficients are enough to approximate f. i.e., $g(x) = \sum_{S s.t. |\hat{f}(S)| \geq \varepsilon/2L_1(f)} \hat{f}(S)\chi_S(x)$ is s.t.* $\text{Exp}[(f-g)^2] \leq \varepsilon/2$

As a result of these claims (which we will prove), we can run the GL algorithm with threshhold parameter $\varepsilon/2L_1(f)$ to obtain a list L of large coefficients. However, there are two apparent difficulties with this approach. (i) GL returns only the *identity* of the large Fourier coefficients, but not their actual values, and (ii) we do not know the value of $L_1(f)$.

To solve the first problem, we will again use sampling to estimate the Fourier coefficients (of the sets $S \subseteq [n]$ which GL returns).

To circumvent the latter problem, we can run GL with a sequence of decreasing parameters: $\left\{\varepsilon/2\tilde{L_1^i}(f)\right\}_{i=0}$, where $\tilde{L_1^i} = 2^i$, stopping after obtaining a function h s.t. $\Pr_x[f(x) \neq sgn(h(x))] \leq \varepsilon$ (to be determined by sampling x). By Claim 8, after at most $\log(m)$ tries, we will have $\tilde{L_1^i} \geq L_1(f)$. In our exposition of the algorithm, we will assume for the sake of simplicity that $\tilde{L_1^i} = L_1(f)$

Algorithm

Let $\tau = \varepsilon/2L_1(f)$, $\delta = \sqrt{\frac{\varepsilon}{2|L|}}$. (Note: $|L| = \text{poly}(\frac{n}{\tau})$)

Use GL to find L, the list of all S s.t. $|\hat{f}(S)| \geq \tau$.
Use sampling to compute $\tilde{f}(S)$ within error δ of $\hat{f}(S)$.
Return $h(x) = \sum_{S \in L} \tilde{f}(S)\chi_S(x)$.

We need $O(\frac{k}{\delta^2}) = O(\frac{2\text{poly}(\frac{n}{\tau})}{\varepsilon}) = O(\text{poly}(\frac{nm}{\varepsilon}))$ samples to estimate each $\tilde{f}(S)$ within δ of $\hat{f}(S)$ with confidence $1 - e^{-k}$.

Now, let $g(x) = \sum_{S s.t. |\hat{f}(S)| \geq \tau} \hat{f}(S)\chi_S(x)$.

Then,

$$\begin{aligned}
\Pr_x[h(x) \neq f(x)] &\leq \Pr_x[h(x) \neq g(x)] + \Pr_x[g(x) \neq f(x)] \\
&\leq \text{Exp}[(h(x) - g(x))^2] + \text{Exp}[(g(x) - f(x))^2] \\
&\leq \sum_{S \in L} (\hat{h}(S) - \hat{g}(S))^2 + \varepsilon/2 \\
&= \sum_{S \in L} (\tilde{f}(S) - \hat{f}(S))^2 + \varepsilon/2 \\
&\leq \sum_{S \in L} \delta^2 + \varepsilon/2 \\
&= |L|\delta^2 + \varepsilon/2 \\
&= \varepsilon/2 + \varepsilon/2 \\
&= \varepsilon
\end{aligned}$$

We restate Claim 9 as a lemma and prove the lemma.

Lemma 10 *For $\varepsilon > 0$, Let $S = \{Z s.t. |\hat{f}(Z)| \geq \frac{\varepsilon}{L_1(f)}\}$, and let $g(x) = \sum_{Z \in S} \hat{f}(Z)\chi_Z(x)$. Then,* $\text{Exp}[(f-g)^2(x)] \leq \varepsilon$

Proof of Lemma 10:

$$(f-g)(x) = \sum_{S \notin S} \hat{f}(Z)\chi_Z(x),$$

so,

$$\begin{aligned}
\text{Exp}[(f-g)^2] &= \sum_{S \notin S} \hat{f}^2(Z) \\
&\leq \max_Z |\hat{f}(Z)| \sum_{Z \notin S} |\hat{f}(Z)| \\
&= \frac{\varepsilon}{L_1(f)} L_1(f) \\
&= \varepsilon
\end{aligned}$$

■

Proof of Claim 8:

For a fixed leaf ℓ, define $g_\ell : \{\pm 1\}^n \to \{0,1\}$ to be

$$g(x) = \begin{cases} 1 & \text{if } x \text{ reaches } \ell \\ 0 & \text{o.w.} \end{cases}.$$

First we will calculate $L_1(g_\ell)$.

Let V_ℓ be the set of indices of variables viewed on the path to ℓ. So if ℓ is of depth k in the tree, then $|V_\ell| = k$ (assuming of course that no variable is seen multiple times on a path down a tree – any function computable by a decision tree is computable by a decision tree that has this property).

To get a sense of the form of g, consider how we can represent g for the leftmost and rightmost leaves:

$$g_{(\ell-left)}(x) = \frac{1-x_{\ell_1}}{2}\frac{1-x_{\ell_2}}{2}\cdots\frac{1-x_{\ell_k}}{2}$$

$$g_{(\ell-right)}(x) = \frac{1+x_{\ell_1}}{2}\frac{1+x_{\ell_2}}{2}\cdots\frac{1+x_{\ell_n}}{2}$$

where ℓ_i is the index of the i^{th} variable seen on the path to ℓ. In general, $g_\ell(x)$ can be represented in this product form, where a $+$ or $-$ is chosen for each factor depending on whether the path to ℓ makes a right or left turn after viewing the corresponding variable. i.e.,

$$\begin{aligned} g_\ell(x) &= \frac{1}{2^{|V_\ell|}}\prod_{i\in V_\ell}(1 \pm x_i) \\ &= \sum_{S\subseteq V_\ell}(\pm 1)\prod_{i\in S} x_i (\text{multivariate polynomial expansion}) \\ &= \frac{1}{2^{|V_\ell|}}\sum_{S\subseteq V_\ell}\chi_S(x) \end{aligned}$$

So,

$$L_1(g_\ell) = \frac{1}{2^{|V_\ell|}}\sum_{S\subseteq V_\ell} = \frac{2^{|V_\ell|}}{2^{|V_\ell|}} = 1$$

Now notice that since f is computable by the decision tree in question, and each x follows exactly one path, we can represent it as follows:

$$f(x) = \sum_\ell g(x)\text{val}(\ell)$$

(where $\text{val}(\ell)$ is the value of the label of l).
Then, by linearity of Fourier coefficients (i.e. $\widehat{f+g} = \hat{f} + \hat{g}$), the triangle inequality, and the fact that $\text{val}(l) \in \{\pm 1\}$, we have for any $T \subseteq [n]$

$$|\hat{f}(T)| = |\sum_\ell \hat{g}_\ell(T)\text{val}(\ell)| \le \sum_\ell |\hat{g}_\ell(T)|$$

Then finally,

$$\begin{aligned} L_1(f) = \sum_T |\hat{f}(T)| &\leq \sum_T \sum_\ell |\hat{g}_\ell(T)| \\ &= \sum_\ell \sum_T |\hat{g}_\ell(T)| \\ &= \sum_\ell L_1(g) \\ &= \sum_\ell 1 \\ &= m \end{aligned}$$

■