COMS 6998-2 Advanced Topics in Computational Learning Theory	**Spring 2005**

Lecture 11 : April 7, 2005

Lecturer: Rocco A. Servedio *Scribe: Ari Polsky and Justin Cranshaw*

1 Lecture Overview

- Our learning methods under the MQ model up to this point have not incorporated any of the complex techniques we have developed with Fourier analysis. Today we will bridge this gap.
- We will give a polynomial time algorithm for learning decision trees with membership queries.
- In addition, we will start to construct a polynomial time algorithm for learning DNF with membership queries.

2 Finding Large Fourier Coefficients

Recall: In the random example model, $EX(f)$, where $f : \{-1,1\}^n \rightarrow \{-1,1\}$ has some $S \subseteq \{x_1 \dots x_n\}$ such that $\left|\hat{f}(S)\right|$ is sufficiently large, we don't have an efficient algorithm for finding S. Using the MQ model, we will show an algorithm for finding S in polynomial time.

Definition 1 *Given* $0 \leq k \leq n$, $S \subseteq \{1,\dots,k\}$, *define*

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

Given $k, S_1, x, EX(f)$ can we efficiently find or approximate $f_{k,S_1}(x)$? The naive way would be to estimate each $\hat{f}(S_1 \cup T_2)$. This will not work becuase there are exponentially many of them. But what we can do is express f_{k,S_1} another way:

$$f_{k,S_1}(x) = E_{y \in \{-1,1\}^k}\left[f(yx)\chi_{S_1}(y)\right] \tag{2}$$

In the $EX(f)$ model, this does not help us, since only examples where the last $n-k$ bits are x are useful, and these occur with probability $\frac{1}{2^{n-k}}$. But in the MQ model, given $k, S_1, x, MQ(f)$ there is a way:

Fact 1 *Given $0 \leq k \leq n, S_1 \subseteq [k], x \in \{-1,1\}^{n-k}, MQ(f)$, we can output $v \in [-1,1]$ such that $\left|v - f_{k,S_1}(x)\right| \leq \gamma$ in time $n \cdot O(\frac{1}{\gamma^2} \cdot log\frac{1}{\delta})$ with probability $1-\delta$*

Proof: Generate $y \in \{-1,1\}^n$ by choosing the first k bits at random. Let the last $n-k$ bits be x. This gives $f(yx)\chi_{S_1}(y) = \pm 1$. Combined with the Chernoff bound, this gives the fact. The extra factor of n comes in becuase we have to do this n times. ■

Now, we will assume for the sake of simplicity that we can exactly compute $f_{k,S_1}(x)$. We can use this to find the $S \subseteq \{x_1 \dots x_n\}$ corresponding to the large Fourier coefficient we are looking for.

Observation 1 *For any $k, S_1 \subseteq \{1 \dots k\}$, we have*

$$E_x\left[f_{k,S_1}(x)^2\right] = \sum_{T_2 \subseteq \{k+1 \dots n\}} \hat{f}(S_1 \cup T_2)^2.$$

This follows directly from Parseval's identity:

$$E\left[f(x)\right]^2 = \sum_{S \subseteq [n]} \hat{f}(S)\chi_S(x)$$

Claim 1 *Let $f : \{-1,1\}^n$, and suppose $\theta > 0$. Then*

1. *At most $1/\theta^2$ many sets S can have $|\hat{f}(S)| \geq \theta$.*
2. *For any value $0 \leq k \leq n$, we have at most $1/\theta^2$ many S_1's such that $E[fk, S_1(x)^2] \geq \theta^2$.*

Proof: (1) This again follows directly from Parseval's identity, since if there were greater than $1/\theta^2$ such S's, then those $\hat{f}(S)^2$ would sum to greater than 1, which is a contradiction.

(2)

$$\begin{aligned} 1 = \sum_{T \subseteq [n]} \hat{f}(T)^2 &= \sum_{S_1 \subseteq [k]} \sum_{T_2 \subseteq \{k+1,\dots,n\}} \hat{f}(S_1 \cup T_2)^2 \\ &= \sum_{S_1 \subseteq [k]} E[f_{k,S_1}(x)^2] \end{aligned}$$

It follows again by Parseval's that at most $\frac{1}{\theta}$ are such that $E[fk, S_1(x)^2] \geq \theta^2$. ■

Now, suppose that we have one large Fourier coefficient. The next claim will be central to our goal of finding this coefficient.

Claim 2 *For $f : \{-1,1\}^n \to \{-1,1\}$, given $k, S_1 \subseteq [k]$, assume that we can obtain the exact value of $E\left[f_{k,S_1}(x)^2\right]$ in 1 time step. Then for any $\theta > 0$ we can identify exactly those S's for which $\left|\hat{f}(S)\right| \geq \theta$ in time poly$(n, \frac{1}{\theta})$.*

Proof: We will do a search over a binary search tree for the correct value of k. The root of the tree will represent $\emptyset$, where $k = 0$. The value stored at each node in the tree is $E\left[f_{k,S}(x)^2\right]$. So at the root, where $k = 0, E\left[f_{k,\emptyset}(x)^2\right] = 1$. The 2 children of the root would represent both possibilities for $k = 1$ - One including 1 in S and one without: The left child would be $E[f_{1,\emptyset}(x)^2]$ and the right child would be $E[f_{1,\{1\}}(x)^2]$, and in general, for the node corresponding to (k, S), its children would be $(k+1, S \cup \{k+1\})$ and $(k+1, S)$. We note that $E[f_{k,S}(x)^2] = E[f_{k+1,S}(x)^2] + E[f_{k+1,S\cup\{k+1\}}(x)^2]$, i.e. that the value of a given node is equal to the sum of the values of its children.

Building this tree in a top-down manner, at each node we will query its 2 children for their values. A node is said to be dead if its value is $< \theta$, and the recursion for that node will terminate, as we are only interested in those nodes with values $\geq \theta$. So the recursion will terminate either with a dead node or a live leaf that has no children.

At step k in this procedure, we know by the earlier claim that there are at most $\frac{1}{\theta^2}$ many live nodes. So in total, since there are n levels in the tree, there are $\frac{n}{\theta^2}$ many live nodes in the tree. At level $k = n$ in the tree, the remaining live leaves exactly correspond to those sets S for which $\left|\hat{f}(S)\right| \geq \theta$. ■

For all this to work, recall that we are relying on the assumption that we can exactly calculate $E\left[f_{k,S_1}(x)^2\right]$. We can't compute this value exactly, but we can approximate it well enough for our purposes. Recall that we can estimate $f_{k,S_1}(x)$ for some x. So all we have to do to get the required expected value is take some random samples of x, estimate $f_{k,S_1}(x)$, compute the average, and square it. Note that becuase $f_{k,S_1}(x) \in [-1,1]$ and hence $f_{k,S_1}(x)^2 \in [0,1]$, we need relatively few samples of x to get an accurate estimate for $E\left[f_{k,S_1}(x)^2\right]$.

This gives us the following theorem:

Theorem 1 (KM) *For any $f : \{-1,1\}^n \to \{-1,1\}, \theta > 0$, there is a poly$(n, \frac{1}{\theta}, log\frac{1}{\delta})$ time algorithm that, with probability $1-\delta$, ouputs a set $\mathcal{S}$ such that:*

- *For every* $S \in \mathcal{S}$, $\left|\hat{f}(S)\right| \geq \theta$
- *For every* $S \notin \mathcal{S}$, $\left|\hat{f}(S)\right| \leq \frac{\theta}{2}$

In other words, $\mathcal{S}$ includes every S such that $\left|\hat{f}(S)\right| \geq \theta$. Note that $|\mathcal{S}| \leq \frac{4}{\theta^2}$.

Theorem 2 *For* $f : \{-1,1\}^n \to \{-1,1\}$*, let* $\mathcal{S}$ *be a set* $\{S_1, S_2 \ldots\}$ *of subsets of* $[n]$ *such that* $\sum_{S \in \mathcal{S}} \hat{f}(S)^2 \geq 1 - \varepsilon$*. There is an algorithm A, that when given* $\mathcal{S}, EX(f), \delta, \tau > 0$*, runs in* $poly(n, |\mathcal{S}|, \frac{1}{\tau}, log\frac{1}{\delta})$ *time and outputs a real-valued function* $g(x) = \sum_{S \in \mathcal{S}} C_S \chi_S(x)$ *such that* $E\left[(g(x) - f(x))^2\right] \leq \varepsilon + \tau$ *with probability* $1 - \delta$*. It also follows that* $Pr_x\left[sign(g(x)) \neq f(x)\right] \leq \varepsilon + \tau$*.*

Proof: The proof is analagous to that of the Low-Degree Algorithm. ■

So, using this theorem, we can learn functions for which there is a small set of large Fourier coefficients that hold most of the power spectrum of f. The extreme case of of a function that has a highly concentrated Fourier spectrum is *PARITY*, or χ_S. For a worst case, consider $\mathrm{IP}_2(x) = (x_1 \wedge x_2) \oplus (x_3 \wedge x_4) \oplus \cdots \oplus (x_{n-1} \wedge x_n)$. This function has every Fourier coefficient $|\hat{f}(S)| = \frac{1}{2^{n/2}}$, since we know that the Fourier coefficient for each $x_i \wedge x_j$ is $\frac{1}{2}(1 + x_1 + x_2 - x_1 x_2)$. To compute the parity of all of these we just multiply terms. Thus the Fourier representation of IP_2 is $\mathrm{IP}_2 = \sum_S \frac{1}{2^{n/2}}(\pm 1 \cdot \chi_S(x))$.

3 Ramifications For Learning

Definition 2 *Given a real-valued boolean function* $f : \{-1,1\}^n \to \mathbb{R}$*, define the L-1 norm of f:*

$$L_1(f) = \sum_{S \subseteq [n]} \left|\hat{f}(S)\right|$$

Intuitively, the L-1 norm of a function measures how spread out its Fourier spectrum is. For example:

$$L_1(\chi_S) = 1$$

$$L_1(IP_2) = 2^{\frac{n}{2}}$$

Theorem 3 *Given $f : \{-1,1\}^n \to \{-1,1\}$ and $\varepsilon > 0$, define*

$$\mathcal{S}_\varepsilon = \left\{ S \subseteq [n] : \left|\hat{f}(S)\right| \geq \frac{\varepsilon}{L_1(f)} \right\}$$

Then:

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

Proof:

1. Since each $S \in \mathcal{S}$ has $\left|\hat{f}(S)\right| \geq \frac{\varepsilon}{L_1(f)}$ and $\sum_{S\in\mathcal{S}} \left|\hat{f}(S)\right| = L_1(f)$, it must be that

$$|\mathcal{S}_\varepsilon| \leq \frac{L_1(f)}{\frac{\varepsilon}{L_1(f)}} = \frac{L_1(f)^2}{\varepsilon}$$

2. $$\sum_{S\notin\mathcal{S}_\varepsilon} \hat{f}(S)^2 \leq \underbrace{\max_{S\notin\mathcal{S}_\varepsilon} \left\{\left|\hat{f}(S)\right|\right\}}_{<\frac{\varepsilon}{L_1(f)}} \cdot \underbrace{\sum_{S\notin\mathcal{S}} \left|\hat{f}(S)\right|}_{\leq\sum_S\left|\hat{f}(S)\right|=L_1(f)} \leq \varepsilon$$

■

Theorem 4 *Combining the KM algorithm with the LMN algorithm, we can learn any $f : \{-1,1\}^n \to \{-1,1\}$ to accuracy ε in time $poly(n, \frac{1}{\varepsilon}, L_1(f), log\frac{1}{\delta})$*

Proof: Run the KM algorithm as before to find $\mathcal{S}_\varepsilon$, then use LMN to get $g(x)$. Output $sign(g(x))$. To run the KM algorithm, you need to know $L_1(f)$, which is unknown. What we can do, though, is keep trying different values and testing them. Try $1,2,4,8\ldots$ until the algorithm produces a hypothesis that works well. ■

Theorem 5 *Let $f : \{-1,1\}^n \to \{-1,1\}$ be a boolean function. If f has an $S-leaf$ DT, $L_1(f) \leq S$.*

Proof: For some fixed leaf ℓ, consider $g_\ell : \{-1,1\}^n \rightarrow \{0,1\}$:

$$g_\ell(x) = \begin{cases} 1 & \text{if x reaches leaf } \ell \\ 0 & \text{otherwise} \end{cases}$$

Without loss of generality, assume ℓ is the left-most leaf in the tree. The path to ℓ would have all variables set to -1. Then:

$$g_\ell(x) = \left(\frac{1-x_1}{2}\right)\left(\frac{1-x_2}{2}\right)\cdots\left(\frac{1-x_k}{2}\right)$$

$$g_\ell(x) = \sum_{S\subseteq\{1\ldots k\}} \frac{1}{2^k}\left(\pm\chi_S(x)\right), \text{ so } L_1\left(g_\ell(x)\right) = \frac{2^k}{2^k} = 1$$

So we have that

$$f(x) = \pm g_{\ell_1}(x) \pm g_{\ell_2}(x) \ldots \pm g_{\ell_S}(x)$$

where $\ell_1 \ldots \ell_S$ are leaves of the DT. Then the Fourier representation of f is

$$\sum_{T\subseteq[n]} \hat{f}(T)\chi_T = \sum_{i=1}^{S} \underbrace{\left(\sum_T \hat{g}_{\ell_i}(T)\chi_T\right)}_{g_{\ell_i}}$$

where

$$\hat{f}(T) = \sum_{i=1}^{S} \pm\hat{g}_{\ell_i}(T) \text{ and } \left|\hat{f}(T)\right| \leq \sum_{i=1} S\left|\hat{g}_{\ell_i}(T)\right|.$$

But then

$$L_1(f) = \sum\left|\hat{f}(T)\right| \leq \sum_T \sum_{i=1}^{S} \left|\hat{g}_{\ell_i}(T)\right| = \sum_{i=1}^{S} \underbrace{\sum_T \left|\hat{g}_{\ell_i}(T)\right|}_{=1} = S.$$

■

3.1 Learning Disjoint DNFs

Definition 3 *An $S-term$ DNF $F = T_1 \vee T_2 \vee \ldots \vee T_S$ is said to be disjoint if every input (assignment to the variables) satisfies at most 1 term.*

Every *size* $-S$ DT has an equivalent disjoint DNF. Let each path from the root to a leaf with a value of 1 be a term in the DNF. It is trivially the case that all such paths are disjoint, and the corresponding DNF is therefore disjoint.

Observation 2 *The same algorithm for learning DT's can be applied to disjoint DNFs.*

If some boolean function $f : \{-1,1\}^n \rightarrow \{-1,1\}$ can be represented as an $S-term$ disjoint DNF, F, then $L_1(f) = O(S)$. Convert F to its binary representation, then change every $\vee$ to a $+$, and you have the Fourier representation of f. So using the algorithm we have seen, we can learn any such disjoint DNF in $poly(n, S, \frac{1}{\varepsilon})$ time. Note that not every $S-term$ DNF has $L_1(f) \leq O(S)$. This is only guaranteed for disjoint $S-term$ DNFs.