6.858/18.428 Machine Learning	**Lecture 22: November 30, 1994**
Lecturer: Ron Rivest	*Scribe: Rebecca Hwa*

22.1 Outline

- Finish up Learning Decision Trees using the Fourier Spectrum with Respect to Uniform Distribution
- Learning DNF with Membership Queries with Respect to Uniform Distribution

22.2 Learning Decision Trees using the Fourier Spectrum

In this lecture we continue the discussion on Kushilevitz and Mansour's polynomial time algorithm for learning decision trees with respect to the uniform distribution using membership queries.

22.2.1 Approximation by sparse functions

We define the basis function χ_z for each $z \in \{0,1\}^n$:

$$\chi_z(x) = (-1)^{z \cdot x \bmod 2}$$

Let $f : \{0,1\}^n \to \{0,1\}$ be the boolean function that represents the target decision tree. To express f as a Fourier series, we expand it as a linear combination of the χ_z's:

$$f(x) = \sum_{z \in \{0,1\}^n} \hat{f}(z)\chi_z(x)$$

where the Fourier coefficients are:

$$\hat{f}(z) = E[f\chi_z].$$

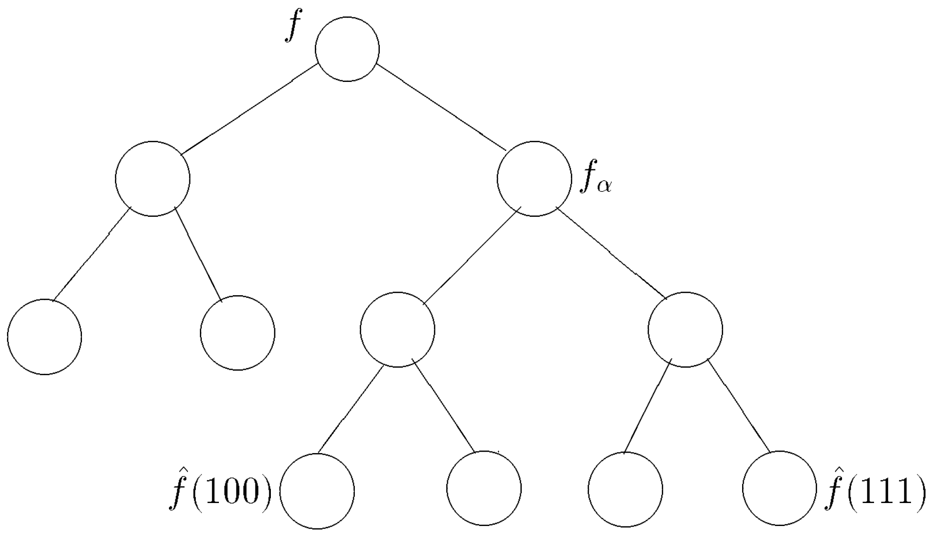

Figure 22.1: the coefficient tree.

We can visualize the Fourier coefficients of f as being arranged on a complete binary tree of depth n, where each path down the tree from the root to a leaf node represents some vector $z \in \{0,1\}^n$, and the leaf node represents the coefficient $\hat{f}(z)$. We label each node in the tree by the path vector from the root to it. Let $\alpha \in \{0,1\}^k$ be the label of an internal node at depth k. (See Figure 22.1). We define a function at node α, which is $f_\alpha : \{0,1\}^{n-k} \rightarrow \Re$:

$$f_\alpha(x) = \sum_{\beta \in \{0,1\}^{n-k}} \hat{f}(\alpha\beta)\chi_\beta(x)$$

So the coefficients of $f_\alpha(x)$ are all the coefficients of f that start with α. The process of finding large coefficients for f, then, can be broken down into finding the large coefficients of the left and right subtrees recursively.

Lemma 1 *Let f be a boolean function, and $\theta > 0$. Then,*

1. *At most $1/\theta^2$ of z satisfy $|\hat{f}(z)| \geq \theta$.*

2. *At any level of the tree at most $1/\theta^2$ of the nodes have $E[f_\alpha^2] \geq \theta^2$.*

Proof: By Parseval's Identity and remembering that f is a boolean function whose coefficients sum to 1, we have:

$$E[f^2] = \sum_{z \in \{0,1\}^n} \hat{f}^2(z) = 1$$

It then follows that there are at most $1/\theta^2$ large coefficients (i.e., $|\hat{f}(z)| \geq \theta$). Similarly, for f_α:

$$E[f_\alpha^2] = \sum_{\beta \in \{0,1\}^{n-k}} \hat{f}^2(\alpha\beta)$$

But if $|\hat{f}(\alpha\beta)| \geq \theta$ for some $\beta \in \{0,1\}^{n-k}$, then $E[f_\alpha^2] \geq \theta^2$. By the above equalities we get

$$\begin{aligned} & \sum_{\alpha \in \{0,1\}^k} E[f_\alpha^2] \\ = & \sum_{\alpha \in \{0,1\}^k} \sum_{\beta \in \{0,1\}^{n-k}} \hat{f}^2(\alpha\beta) \\ = & \sum_{z \in \{0,1\}^n} \hat{f}^2(z) = 1 \end{aligned}$$

So at most $1/\theta^2$ nodes have $E[f_\alpha^2] \geq \theta^2$. ■

We now present the algorithm `Coef`(α). The initial call is `Coef`(λ), where λ is the empty string.

```
Subroutine Coef(α)
    if E[f_α^2] ≥ θ^2 then
        if |α| = n then output α
            else Coef(α0);Coef(α1)
```

Corollary 1 `Coef` *examines at most n/θ^2 nodes of the tree (i.e., it's invoked at most n/θ^2 times).*

Proof: From lemma 1 we know that the number of α's for which $E[f_\alpha^2] \geq \theta^2$ is at most $1/\theta^2$, for each length of α. So the total number of recursive calls is in $O(n/\theta^2)$. ■

The algorithm calls for the exact value of $E[f_\alpha^2]$ which is computationally costly. The good news is, we can estimate $E[f_\alpha^2]$ accurately in polynomial time—with an error less than $\theta^2/4$ with very high probability. To use the approximated values requires a minor change in the algorithm described above. Instead of testing on $(E[f_\alpha^2] \geq \theta^2)$, we will test on whether the approximate value of $E[f_\alpha^2]$ is greater than $\theta^2/2$.

To approximate $E[f_\alpha^2]$ we approximate the value of f_α. Remembering from the last lecture that $f_\alpha(x) = E_{y\in\{0,1\}^k}[f(yx)\chi_\alpha(y)]$, we can then approximate f_α by picking m_1 values of x, where $|x| = n-k$ and picking m_2 values of y, where $|y| = k$. To evaluate $f(yx)\chi_\alpha(y)$ for each pair of (x, y), we make queries for each $f(yx)$. The size of m_1 and m_2 can be determined by Chernoff bounds:

$$\begin{aligned} m_1 &= O\left(1/\theta^2 \ln\left(\frac{n}{\theta^2\delta}\right)\right) \\ m_2 &= O\left(1/\theta^2 \ln\left(\frac{m_1 n}{\theta^2\delta}\right)\right) \end{aligned}$$

22.2.2 Functions with small L_1 norm

We now turn our attention to functions with small L_1 norms; i.e.,

$$L_1(f) = \sum_z |\hat{f}(z)|,$$

where $1 \leq L_1(f) \leq 2^{n/2}$ for some boolean function f. The L_1 norm measures how spread out things are. A function with a small norm has a small number of large coefficients. In the next lemma, we show that approximating the coefficients of f is good enough for an approximation of f.

Lemma 2 *Given $\epsilon > 0$, let $S = \{z : |\hat{f}(z)| \geq \epsilon/L_1(f)\}$, and let $g(x) = \sum_{z\in S}\hat{f}(z)\chi_z(x)$. Then*

$$E[(f-g)^2] \leq \epsilon.$$

This implies that

$$Prob[f(x) \neq sign(g(x))] \leq \epsilon.$$

Proof: By the definition of g, we have

$$(f-g)(x) = \sum_{z \notin S} \hat{f}(z)\chi_z(x)$$

By Parseval's identity, we have

$$E[(f-g)^2] = \sum_{z \notin S} \hat{f}^2(z).$$

The upper bound is

$$\begin{aligned} & \mathrm{Max}_{z \notin S}|\hat{f}(z)| \cdot \left(\sum_{z \in \{0,1\}^n} |\hat{f}(z)| \right) \\ \leq \quad & \frac{\epsilon}{L_1(f)} \cdot L_1(f) = \epsilon. \end{aligned}$$

■

The above lemma implies that the threshold value θ is $\epsilon/L_1(f)$ and f can be approximated by the coefficients whose absolute values are greater than the threshold value. Thus we can approximate f by using procedure `Coef` to find all coefficients that are greater in absolute value than $\epsilon/L_1(f)$.

22.2.3 Decision trees

Now let us consider the decision tree that the boolean function f is representing. In the next lemma, we show that the L_1 norm of f is bounded by m, the number of leaves in the decision tree.

Lemma 3 *Let f be computed by a decision tree with m nodes, then $L_1(f) \leq m$.*

Proof: Let $\texttt{leaf}(T_f)$ be the set of leaves in the decision tree T_f, and let $d(v)$ be the number of nodes on the path from the root to node v, but not including v. For each node at depth d, exactly $1/2^{-d}$ fraction of the inputs reaches it. Let $I(v)$ be the set of all the inputs that reach leaf v. Then for every z,

$$\begin{aligned} & \hat{f}(z) = E[f\chi_z] \\ = \quad & \sum_{v \in \texttt{leaf}(T_f)} 2^{-d(v)} val(v) E_{x \in I(v)}[\chi_z(x)]. \end{aligned}$$

Kushilevitz and Mansour show that $|E_{x\in I(v)}[\chi_z(x)]| = 1$ for exactly $2^{d(v)}$ values of z and 0 for the rest. Thus, each leaf v contributes to at most $2^{d(v)}$ coefficients $\hat{f}(z)$, and to each such coefficient, it contributes at most $2^{-d(v)}$. Thus, leaf v contributes at most one to the sum of the absolute value of all the coefficients. So $L_1(f) \leq$ number of leaves $\leq$ total number of nodes. More formally:

$$
\begin{aligned}
L_1(f) &= \sum_{z\in\{0,1\}^n} |E[f\chi_z]| \\
&= \sum_{z\in\{0,1\}^n} \left| \sum_{v\in\mathtt{leaf}(T_f)} 2^{-d(v)} val(v) E_{x\in I(v)}[\chi_z] \right| \\
&\leq \sum_{z\in\{0,1\}^n} \sum_{v\in\mathtt{leaf}(T_f)} 2^{-d(v)} \left| E_{x\in I(v)}[\chi_z] \right| \\
&= \sum_{v\in\mathtt{leaf}(T_f)} 2^{-d(v)} \sum_{z\in\{0,1\}^n} \left| E_{x\in I(v)}[\chi_z] \right| \\
&= \sum_{v\in\mathtt{leaf}(T_f)} 2^{-d(v)} 2^{d(v)} \\
&\leq |\mathtt{leaf}(T_f)| \leq m
\end{aligned}
$$

■

Now we have finished showing that there is a polynomial time algorithm that for any boolean function f that can be represented by an m node decision tree with linear operations, and for any $\epsilon, \delta > 0$, outputs a function g such that with probability at least $1-\delta$, $\Pr[f \neq sign(g)] \leq \epsilon$, and the algorithm runs in time polynomial in n, m, $1/\epsilon$, and $\log 1/\delta$.

22.3 Learning DNF with Membership Queries

We now briefly discuss Jackson's algorithm for learning DNF with respect to the uniform distribution. This algorithm is based on two ideas that we've seen in the course. One is Freund's boosting technique to boost the weak learner into a strong learner. The other is Kushilevitz and Mansour's decision tree learning algorithm that we saw in the previous sections.

The hypothesis boosting algorithm requires a vote from a set of functions $g_i(x)$, where $1 \leq i \leq k$. We let

$$g_1(x) = \hat{f}(z)\chi_z(x),$$

but we will need to have non-uniform distributions for $g_2(x), \ldots, g_k(x)$.

Fact 1 *For every DNF f with s terms and for every distribution D on the instance space of f there exists a χ_A such that $|E_D[f\chi_A]| \geq 1/(2s+1)$.*

We need an oracle that, when given x, computes $D(x)$. We choose $g(x) = 2^n f(x)D(x)$. We can then simulate an oracle for this g given oracles for f and D.

$$(\forall A) E_D[f\chi_A] = E[g\chi_A] = \hat{g}(A).$$

Theorem 1 *DNF is learnable with respect to the uniform distribution using membership queries.*

Corollary 2 *Every DNF f with s terms can be computed as the majority of $O(ns^2)$ parity functions.*

References

[1] Jeffrey Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 42-53, Santa Fe, New Mexico, November 1994. IEEE.

[2] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, pages 455-464, New Orleans, Louisiana, May 1991. ACM.