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

Lecture 7 : March 3, 2005

Lecturer: Rocco A. Servedio *Scribe: Bert Huang*

1 Overview

In this lecture we cover the following:

- Learning DNF in quasi-polynomial time under the Uniform Distribution $\mathcal{U}$
- Fourier analysis on $\{-1,1\}^n$

2 Learning DNF

Here we explain an algorithm for PAC learning DNF in quasi-polynomial time under the Uniform Distribution $\mathcal{U}$.

2.1 Limiting the length of the terms

Consider any s-term DNF f such that $f = T_1 \vee \ldots \vee T_s$, where $T_1, \ldots, T_s$ are conjunctions of any length.

Fact 1 *For $f = T_1 \vee \ldots \vee T_s$, let f' be f, but with all terms of length $> \log \frac{2s}{\tau}$ removed, then $Pr_{x\in\mathcal{U}}[f(x) \neq f'(x)] \leq \frac{\tau}{2}$.*

Proof: Any given term of length k is satisfied with probability $\frac{1}{2^k}$, so any term discarded is satisfied with probability $\leq \frac{\tau}{2s}$. Since $\leq s$ terms can be discarded, the probability one of these long terms will affect the output of $f(x)$ is at most $\frac{\tau s}{2s} = \frac{\tau}{2}$. ∎

2.2 The Return of Winnow

Recall the Winnow algorithm, which is an online learner for linear threshold functions. It learns weight-W LTFs over $\{0,1\}^n$ with a $p(W, \log n)$ mistake bound, where p is polynomial.

In fact, when Winnow is used to learn a disjunction of k unknown variables, its mistake bound is $O(k \log n)$. (This is a classic computational learning theory result, covered in the intro course, 4252)

Recall the fact from lecture 6 that we can convert any online learning algorithm into a PAC learner. We will convert Winnow to a PAC learning algorithm, resulting in a PAC version that learns any length-k disjunction, under any distribution over $\{0,1\}^n$, using $\frac{O(k \log n)}{\epsilon}$ many examples.

2.3 The Algorithm

- To simplify notation, let us name our cutoff for term length we defined in section 2.1, $\log \frac{2s}{\tau} = r$
- Given example $x \in \{0,1\}^n$, we'll expand x into a vector of values of all possible (length $\leq r$)-conjunctions on x. This vector will be an N-dimensional vector, where $N = \sum_{i=1}^{r} 2^i \binom{n}{i} \approx n^r$. Let us call this expanded vector $x_{expanded}$
- Let $\mathcal{D}$ be the distribution this induces over $\{0,1\}^N$. Note that this is not uniform.
- The DNF f' is a disjunction of $k \leq s$ of these variables over N-dimensional space.
- If we can get $O(\frac{k \log n}{\epsilon}) = M \ll \frac{sr \log n}{\epsilon}$ examples from the oracle $EX(f', \mathcal{D})$, then Winnow can be used to get a hypothesis h such that $Pr_{x \in \mathcal{D}}[h(x) \neq f'(x)] \leq \frac{\epsilon}{2}$ (we scale ϵ by a constant at this stage to simplify the result, since we are dealing with big-O notation anyway)
- This probability can be rewritten as $Pr_{x \in \mathcal{U}}[h(x_{expanded}) \neq f'(x)] \leq \frac{\epsilon}{2}$
- Recall from section 2.1 that $Pr_{x \in \mathcal{U}}[f(x) \neq f'(x)] \leq \frac{\tau}{2}$
- Thus, $Pr_{x \in \mathcal{U}}[h(x_{expanded}) \neq f(x)] \leq \frac{\tau}{2} + \frac{\epsilon}{2}$, which is $\leq \epsilon$ if $\tau \leq \epsilon$

With this, we now know that if we can get M examples from $EX(f', \mathcal{D})$, then this learner will succeed. It turns out that if $\frac{\tau}{2}$ is much less than $\frac{1}{M}$, we will, with high probability, get only examples from $EX(f', \mathcal{D})$.

Since $\frac{1}{M} = \frac{\epsilon}{sr \log n}$, the condition $\frac{\tau}{2} \ll \frac{1}{M}$ subsumes the previously stated condition, $\tau \leq \epsilon$, so we only need to worry about the first one.

We can pick $\tau = \frac{1}{100} \frac{\epsilon}{sr \log n}$.

With this τ, $\log \frac{2s}{\tau} = \log(\frac{200 s^2 r \log n}{\epsilon})$. We should confirm that this is less than r.

If we assume $s \geq \log n$, we can take $r = 10 \log(\frac{s}{\epsilon}) = \log(\frac{s^{10}}{\epsilon^{10}})$

We want $\frac{s^{10}}{\epsilon^{10}} \geq \frac{200 s^2 r \log n}{\epsilon} = \frac{200 s^2 10 \log(\frac{s}{\epsilon}) \log n}{\epsilon}$

Divide both sides by $\frac{s^2}{\epsilon}$ to get $\frac{s^8}{\epsilon^9} \geq 2000 \log(\frac{s}{\epsilon}) \log n$

Given that $s \geq \log n$, we know we can divide out another s^2 from both sides and get $\frac{s^6}{\epsilon^9} \geq 2000$, which is easily satisfied by any meaningful s and ϵ.

We've thus sketched a proof of:

Theorem 1 *For $s \geq \log n$, this algorithm learns s-term DNF over $\{0,1\}^n$ in time $n^{O(\log(\frac{s}{\epsilon}))}$ using poly$(s, \frac{1}{\epsilon})$ examples.*

Note: this bound is much smaller than our bound for online learning of DNF, which was an $n^{O(n^{\frac{1}{3}} \log s)}$-time algorithm. Intuitively, we're able to exploit the fact that we only need to be approximately correct rather than exactly right.

2.4 Further Reading

(Taken from the official course website) For a different approach to learning DNF under uniform in quasipolynomial time (based on greedy set cover rather than Winnow), see Verbeurgt's COLT (Conference on Computational Learning Theory) 1988 paper (you can find COLT 1988 in the Kosoresow memorial library on the second floor of the CS department).

3 Fourier Analysis

From now on, to make our analysis cleaner, we will treat the boolean cube as $\{-1, 1\}^n$. We will also view boolean functions as a subset of real value functions, rather than a seperate class. This means while we normally view boolean functions as $\{-1, 1\}^n \to \{-1, 1\}$, we will now treat them as $\{-1, 1\}^n \to \mathbb{R}$, just with outputs that happen to be -1 or 1.

3.1 Vector Representations of Functions

A real valued function, $f : \{-1, 1\}^n \to \mathbb{R}$, can be viewed as a 2^n-dimension real vector, where the indices are the enumeration of all 2^n possible inputs. For example, the following is a real-valued function f on 2 inputs:

$f(-1, -1) = 3$

$f(-1, 1) = 2$

$f(1, -1) = 5$

$f(1, 1) = 4$

So f can be viewed as the vector $(3, 2, 5, 4)$.

We will write V for the 2^n-dimensional vector space of all $f : \{-1, 1\}^n \to \mathbb{R}$.

Given any $v = (v_1, \ldots, v_{2^n}) \in V$, we define $v = v_1e_1 + v_2e_2 + \ldots + v_{2^n}e_{2^n}$, where $v_1, \ldots, v_{2^n}$ are real numbers and the e's are simple canonical basis vectors:

$e_1 = (1, 0, \ldots, 0)$

$e_2 = (0, 1, \ldots, 0)$

...

This decomposition of v corresponds to the usual way of looking at a function by considering each of its "output values".

3.2 Fourier Representation of f

We can also write f as a linear combination of elements of a different basis: 2^n parity functions.

Definition 1 *Let* $[n] = \{1, \ldots, n\}$. *Given a subset* $S \subseteq [n]$, *the parity corresponding to* S *is*

$$PAR_S(x_1, \ldots, x_n) = \begin{cases} -1 & \text{if an odd number of } i \in S \text{ have } x_i = -1 \\ +1 & \text{if an even number of } i \in S \text{ have } x_i = -1 \end{cases}$$

Note that these parity functions can be written as monomials.

$PAR_S(x_1, \ldots, x_n) \equiv \prod_{i \in S} x_i$

To simplify notation, we will name $PAR_S = \chi_S$ (and we may sometimes write just x_S to denote χ_S).

For example: if $n = 2$, the 4 parities are:

(x_1, x_2)	$(1, 1)$	$(1, -1)$	$(-1, 1)$	$(-1, -1)$
$\chi_\emptyset$	1	1	1	1
$\chi_{\{1\}}$	1	1	-1	-1
$\chi_{\{2\}}$	1	-1	1	-1
$\chi_{\{1,2\}}$	1	-1	-1	1

We will show later that these vectors are linearly independent.

Definition 2 *The inner product on* $\mathbb{V}$ *is defined as follows: Given* $f, g \in \mathbb{V}$,

$$< f, g >= \frac{1}{2^n} \sum_{x \in \{-1,1\}^n} f(x)g(x).$$

This is also equal to the expected value, $E_{x \in \mathcal{U}}[f(x)g(x)]$; this viewpoint will be very useful.

Since we have an inner product, there is a corresponding associated norm:

$$||f|| = \sqrt{< f, f >} = \sqrt{\frac{1}{2^n} \sum_{x \in \{-1,1\}^n} f(x)^2} = \frac{1}{2^{n/2}} \sqrt{\sum_{x \in \{-1,1\}^n} f(x)^2}.$$

3.3 Properties of Parity functions

1. $||\chi_S|| = 1$

Proof: This is not only true for parity functions, but it is also true for all boolean functions. Consider the definition of the norm,
$\frac{1}{2^{n/2}}\sqrt{\sum_{x\in\{-1,1\}^n} f(x)^2}$. Since $1^2 = (-1)^2 = 1$, then $\frac{1}{2^{n/2}}\sqrt{\sum_{x\in\{-1,1\}^n} 1} = \frac{1}{2^{n/2}}\sqrt{2^n} = 1$ ■

2. For all $S, T \subseteq [n]$, $\chi_S \chi_T \equiv \chi_{S\Delta T}$

   (Δ is the symmetric difference of two sets: the set of elements in exactly one of the two sets.)

   Proof:

   Example: $\chi_{\{1,3,4\}}\chi_{\{3,4,6\}} = (x_1x_3x_4)(x_3x_4x_6) = (x_1x_6)(x_3^2x_4^2) = (x_1x_6) \times 1$

   Each variable that appears in both S and T will multiply itself, resulting in 1, so the result is the parity of the remaining variables, the symmetric difference.

   ■

3. If $S = \emptyset$, $E[\chi_S] = 1$. If $S \neq \emptyset$, $E[\chi_S] = 0$

   Proof:

   For $S = \emptyset$, it is trivial that $\chi_S(x) \equiv 1$ for all x.

   When $S \neq \emptyset$, exactly 2^{n-1} of the 2^n inputs have $\chi_S(x) = 1$.

   We can see this by envisioning 2^{n-1} pairs of inputs such that inputs in a given pair are exactly identical except for one bit. Then each pair will contain exactly one input that has an even number of -1's and exactly one input that has an odd number of -1's.

   Thus, 2^{n-1} inputs will output 1 and 2^{n-1} inputs will output -1, and the expected value is 0. ■

4. $<\chi_S, \chi_T> = \begin{cases} 1 & \text{if } S \equiv T \\ 0 & \text{otherwise} \end{cases}$

   This means the PAR functions are orthogonal.

   Proof: $<\chi_S, \chi_T> = \frac{1}{2^n}\sum_{x\in\{-1,1\}^n} \chi_S(x)\chi_T(x)$

   Using property 2, we can convert this to:

   $\frac{1}{2^n}\sum_{x\in\{-1,1\}^n} \chi_{S\Delta T}(x)$

So, using property 3, we can tell that if $S \equiv T$, $S\Delta T = \emptyset$, so the output is 1. Otherwise it will be nonempty, and the output is 0.

$\blacksquare$

5. The set of all 2^n χ_S's are linearly dependent.

   Proof: This follows immediately from orthogonality. To see this, recall that if the χ_S's are linearly dependent, then there is some nonzero linear combination $c_{S_1}\chi_{S_1} + c_{S_2}\chi_{S_2} + \ldots$ that equals the zero vector. But now if we take inner product with χ_{S_1}, we get:

   $< \vec{0}, \chi_{S_1} >= 0 = c_{S_1} < \chi_{S_1}, \chi_{S_1} > + c_{S_2} < \chi_{S_2}, \chi_{S_1} > + \ldots$

   Note that by property 4, we know the term in the first element of the right side of this equation, $< \chi_{S_1}, \chi_{S_1} >= 1$, while the rest of the terms, such as $< \chi_{S_2}, \chi_{S_1} >= 0$. This shows that the constant c_{S_1} must in fact be 0. The same reasoning shows that each coefficient c_{S_i} equals 0, but this contradicts the fact that we had a nonzero linear combination. $\blacksquare$

3.4 Fourier Coefficients

We now know that any $f : \{-1,1\}^n \to \mathbb{R}$ has a unique decomposition as a linear combination of these basis elements, parity functions.

$f(x) = \sum_{S\subseteq[n]} c_S \chi_S(x)$

We will write $\hat{f}(S)$ for $c_s : f(x) = \sum_{S\subseteq[n]} \hat{f}(S)\chi_S(x)$

$\hat{f}(S)$ are the Fourier coefficients of f. These are the coefficients of the unique multi-linear polynomial computing f. This is true because χ_S's are a basis.

Fact 2 $\hat{f}(S) =< f, \chi_S >= E[f(x)\chi_S(x)]$

Proof:

$$< f, \chi_S >=< \sum_{T\subseteq[n]} \hat{f}(T)\chi_T, \chi_S >= \sum_{T\subseteq[n]} \hat{f}(T) < \chi_T, \chi_S >= \hat{f}(S)$$

where the last equality comes from Property 4. $\blacksquare$

If f is a boolean function,

$$\hat{f}(S) = E[f(x)\chi_S(x)] = (1)\Pr[f = \chi_S] - (1)\Pr[f \neq \chi_S] = 2p - 1$$

where $p = \Pr[f = \chi_S]$. Note that $\hat{f}(S) \in [-1, 1]$ and that it measures the correlation between f and χ_S, so that

1. $p = 1 \Rightarrow f = \chi_S$ for all x and the coefficient $\hat{f}(S) = 1$
2. $p = 0 \Rightarrow f = -\chi_S$ for all x and the coefficient $\hat{f}(S) = -1$
3. $p = \frac{1}{2} \Rightarrow f = -\chi_S$ half the time and the coefficient $\hat{f}(S) = 0$.

Let's look at an example of the Fourier analysis of a simple boolean function and then explore a useful property of Fourier coefficients.

Example 1 *Fourier analysis of AND function.*

$$f(x) = AND(x_1, \ldots, x_k) = \begin{cases} 1 & \text{if } x_1 = \ldots = x_k = -1 \\ -1 & \text{if any of } x_1, \ldots, x_k \text{ is } 1 \end{cases}$$

Now let $g(x_1, \ldots, x_k) = \frac{1+f}{2}$ so that g is 1 if $x_1 = \ldots = x_k = -1$ and 0 if any $x_i = 1$. Then

$$g(x_1, \ldots, x_k) = \frac{1 - x_1}{2}\frac{1 - x_2}{2} \ldots \frac{1 - x_k}{2} = \sum_{S \subseteq [k]} \frac{(-1)^{|S|}}{2^k}\chi_S$$

where $\frac{(-1)^{|S|}}{2^k}$ are the Fourier coefficients of g. By our definition,

$$f(x) = 2\sum_{S \subseteq [k]} \frac{(-1)^{|S|}}{2^k} - 1 = -1 + \frac{2}{2^k} + \sum_{|S| \geq 1} \frac{2(-1)^{|S|}}{2^k}\chi_S$$

where $-1 + \frac{2}{2^k} = \hat{f}(\emptyset)$.

We now prove a useful general property of Fourier coefficients:

Theorem 2 (Plancherel's Theorem) *For any functions $f, g : \{-1, 1\}^n \to \mathbb{R}$, we have that*

$$\sum_{S \subseteq [n]} \hat{f}(S)\hat{g}(S) = \frac{1}{2^n} \sum_{x \in \{-1,1\}^n} f(x)g(x) = E[fg] =< f, g >$$

Proof:

$$E_x[fg] = E_x[(\sum_{S\subseteq[n]} \hat{f}(S)\chi_S(x))(\sum_{T\subseteq[n]} \hat{g}(T)\chi_T(x))] = \sum_{S\subseteq[n]}\sum_{T\subseteq[n]} \hat{f}(S)\hat{g}(T)E_x[\chi_S(x)\chi_T(x)]$$

Recall that $\chi_S(x)\chi_T(x) = \chi_{S\Delta T}$, so

$$E_x[\chi_S(x)\chi_T(x)] = E_x[\chi_{S\Delta T}(x)] = \begin{cases} 1 & \text{if } S \equiv T \\ 0 & \text{otherwise} \end{cases}$$

which gives us that $E_x[fg] = \sum_{S\subseteq[n]} \hat{f}(S)\hat{g}(S)$. ■

Now this theorem can be applied to boolean functions, giving us the following:

Corollary 1 *For any Boolean function $f : \{-1,1\}^n \to \{-1,1\}$, we have $\sum_{S\subseteq[n]} \hat{f}(S)^2 = E[f \times f] = 1$.*

So for any boolean function f, the sum of squares of its Fourier coefficients is 1.

3.5 Intuitions About Learning

- We may view $\hat{f}(S)^2$ as the amount of "energy" that f has on χ_S, where a Boolean f has one unit of "energy" to spend in total.
- In uniform distribution learning our goal is to get a good approximation of f.
- A good way of learning a boolean function is to find where it *spends most of its energy*; intuitively this should let us come up with a good approximation.
- Next time we'll see that if we can get a good approximation of the "important" Fourier coefficients of f, we can learn f to high accuracy.

3.6 Further Reading

(Taken from the official course website) A very useful source for uniform distribution learning results based on Fourier techniques is the survey by Mansour; you can get it at http://citeseer.ist.psu.edu/111872.html