Analysis of Boolean Functions **(CMU 18-859S, Spring 2007)**

Lecture 11: Learning juntas with Siegenthaler's Theorem

Feb. 20, 2007

Lecturer: Ryan O'Donnell *Scribe: Yi Wu*

1 The problem of learning r-junta

Problem: Let $C_r = \{f : \{-1, 1\}^n$and f is r-junta$\}$, we are given access to uniform random examples and our goal is to learn f with high confidence.

This lecture, we will present an algorithm running in time $n^{0.704r}$.

2 Learning Tools

Below are some tools needed in analyzing the algorithm of learning r-junta.

2.1 Finding a single relevant variable is enough

The following proposition illustrate that we only need to find efficient algorithm that is able to return single relevant variable.

Proposition 2.1 *If there is an algorithm running in time* $n^\alpha \mathrm{poly}(n, 2^r, \log(\frac{1}{\delta}))$ *which can guarantee to find a relevant variable given an* r*-junta or to determine if f is constant. Then we can learn the class* C_r *in the same time.*

Proof: First we can determine if f is constant with probability bigger than $1-\delta$ in time $O(2^r \log(1/\delta))$(we have high probability get all 2^r possible input).

Suppose we have found that coordinate i is relevant variable for f. Consider then two restriction of f:

$$f_{-1\to i}, f_{1\to i}$$

This is some $(r-1)$junta. We can still simulate random access the for this two functions. If we want to draw M examples for one of the function say $f_{-1\to i}$, we can draw $2M\log(1/\delta)$ examples from f and keep ones with $x_i = -1$. Doing that simulation, we can use the black box algorithm to find relevant variable for $f_{-1\to i}, f_{1\to i}$. And we can keep branching on the two relevant variable we find. Essentially, we can construct a tree for f. And the depth of the tree is at most r . Each node of the tree is function f restricted on some set of k $(0 \le k \le r)$ relevant variables. And we can always simulating M random access to that function by $2^k M \log(1/\delta)$ examples. Notice that the black box algorithm run at most 2^r times and each time the example we need to draw is at most 2^r times of the original samples. So the time of finding the r relevant variables is still within time

$n^{\alpha}\mathrm{poly}(n, 2^r, \log(\frac{1}{\delta}))$. After identifying the r relevant variables, we can simply draw $2^r \log(1/\delta)$ variables and with high probability we will see every 2^r input and decide truth table of the r-junta. Overall, this is an $n^{\alpha}\mathrm{poly}(n, 2^r, \log(\frac{1}{\delta}))$ algorithm.□

2.2 Learning low degree fourier expansion

From the previous section, we know in order to learn r-junta, it suffice to find an algorithm to identify relevant variables. If a function has a non-zero degree $\leq d$ term in fourier expansion, we can achieve that goal as following:

We first estimate all fourier coefficients up to degree d with accuracy $\pm 2^d/4$, and then round them to integer of multiplier of 2^d. Notice $\hat{f}(s) = \mathbf{E}_x[f(x)\chi_s(x)]$, we can estimate all the low degree coefficient $\hat{f}(s)$ accurately within time $n^d\mathrm{poly}(n, 2^r, \log(1/\delta))$. We then have the accurate value of the coefficients up to degree d. Notice variable in a non-zero fourier expansion term is relevant. By checking the fourier term, we can identify the relevant variables.

2.3 Learning low degree function on $\mathbb{F}_2$

This section, we will show the algorithm finding relevant variables for function of low degree on $\mathbb{F}_2$.

Proposition 2.2 *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$, then f can be uniquely represented as a multilinear polynomial over F_2.*

Proof: Write down the interpolation as following:

$$f = \sum_{a \in F_2^n} f(a) \prod_{i=1}^{n} (x_i - a_i)$$

Expand it and we will get multi-linear polynomial. □

Example 2.3 *$parity(x_1, x_2 ... x_n)$ is degree 1 in $\mathbb{F}_2$.*

Example 2.4 *$And(x_1 ... x_n) = x_1 x_2 .. x_n$ is degree n in F_2.*

Example 2.5 $x_1 \wedge x_2 .. \wedge x_{d-1} \bigoplus x_{r-d} \bigoplus ... x_r = x_1 x_2 .. x_{r-d-1} + x_{r-d} + ... x_r$

Theorem 2.6 *The class of function $\{f : F_2^n \to F_2, deg_{F_2}(f) \leq e\}$ is learnable for random examples in time $n^{we}\mathrm{poly}(n) log(1/\delta)$.Here ω is the coefficients that you can do $n \times n$ matrix multiplication or inversion in time $n^{\omega + O(1)}$. The best w known currently is 2.376.*

Proof: Here is the sketchy of the proof.
We draw m examples where $\boldsymbol{X} = (\boldsymbol{x}^j, f(\boldsymbol{x}^j))_{j=1...m}$, here $m = n^e O(2^e log(1/\delta))$. We want to find a function p have $degree_{F_2} \leq e$ and it is consistent with the data. By easy learning theory, p is equal to f with high probability. We write down a linear equation for each samples:

$$\sum_{|s|\leq e} c_s \prod_{i\in S} x_i = f(x_i)$$

We view c_s as unknown variables we want to find. There are at most n^e unknown variable c_s in the expansion of f at F_2. So we can solve the problem with a matrix inversion in time $O(n^{ew})$.□

3 Main algorithm for Learning r-junta

3.1 T. Siegenthaler' theorem

Definition 3.1 *$g : \{-1,1\}^r \to \{-1,1\}$ is called dth order immune if $\hat{g}(s) = 0, \forall 0 < |s| < d$.*

Proposition 3.2 *(In homework 1) A function g is dth order immune $\iff$ $\mathbf{E}[g_{X\to I}] = E[g]$ for any restriction $|I| \leq d$.*

Example 3.3 *$(x_1 \bigoplus x_2 .. \bigoplus x_{2r/3}) \wedge (x_{r/3+1} \bigoplus ..x_r)$ is 2r/3 order immune.*

Next theorem from T. Siegenthaler shows that a function is either of low degree in Fourier expansion or of low degree in $\mathbb{F}_2$.

Theorem 3.4 *Let $g : \{T, F\}^r \to \{T, F\}$ be dth order correlation immune. Then the F_2 polynomial for g has degree at most $r - d$.*

Proof: Assume $d < r$, otherwise, g is constant function. So the fourier expansions of g looks like:

$$g_R(x) = \hat{g}(\phi) + \sum_{r>|s|>d} \hat{g}(s)\chi_s(x)$$

Let $h_R = g_R \bigoplus PARITY_{[r]}$, it suffice to show $deg_{F_2}(h) \leq r - d$ because we have in $\mathbb{F}_2$ that $h_{\mathbb{F}_2} = g_{\mathbb{F}_2} + x_1 + ...x_n$.

(a) If $\hat{g}(\phi) = 0$, then fourier expansion of $h(x)$ has degree at most $r - d - 1$. We now show how to convert it into its $\mathbb{F}_2$ form by following procedure:

1. Replace each x_i with $1 - 2x_i$,
2. Halve it and subtract it from $\frac{1}{2}$.
3. Reduce polynomial's coefficient by mod 2.

After step 1 when we replace term like $x_1...x_n$ with $(1-2x_1)(1-2x_2)...(1-2x_n)$ the degree can not go up(or down). At step 2, the degree is unchanged. And the coefficients are integers after step 2 because we can uniquely write h into its multilinear form by adding up terms like $x_1x_2(1-x_3)...f(1,1,0...)$ and the expansion of it only have integer coefficients. At step 3, the degree can only go down(some high degree may be even). So over all, this conversion shows h(hence g) is at most of degree $r-d-1$ in $\mathbb{F}_2$.

(b) If $g(\phi) \neq 0$, we can still do the conversion of h into $\mathbb{F}_2$. Compared with situation (a), we have one more term of the form $\hat{g}(\phi)(1-2x_1)...(1-2x_r)$ after the first step and $-\frac{1}{2}\sum(-2)^s g(\phi)\prod_{i\in s} x_i$ after the second step. Notice that after step 2, all coefficient should be integer. So $-\frac{1}{2}\sum(-2)^{r-d}g(\phi)$ is integer because the other term from case (a) has degree up to $r-d-1$. Then all the term with degree $\geq r-d+1$ is even. Hence they are dropped off when doing the mod 2. So the degree is at most $r-d$ in $\mathbb{F}_2$.

□

3.2 Algorithm of learning k-junta

Given all the preparation, we have our final theorem:

Theorem 3.5 *The class of r-junta over n-bits can be learned under uniform distribution with confidence $1-\delta$, in time $n^{wr/(w+1)}poly(n, 2^r, \log(1/\delta)$.*

Proof: Run the algorithm finding low degree fourier coefficients up to degree $d = wr/(w+1)$, it is within time $n^{wr/(w+1)}$. If no relevant variable found, then g is at most $r-d = r/(w+1)$ degree in $\mathbb{F}_2$. Use the algorithm in Theorem 2.9, we can find relevant variable in $n^{wr/(w+1)}$.□