COMS 6253: Advanced Computational Learning Theory **Spring 2012**

Lecture 10: March 29, 2012

Lecturer: Rocco Servedio *Scribe: Mengqi Zong*

1 Last Time and Today

Previously:

- Use noise sensitivity NS_ϵ to get Fourier concentration
- Proved Peres Theorem on NS_ϵ of halfspaces
- Uniform distribution learning beyond LDA
  - Learning LTFs, the "Chow parameters"
  - Learning random DNFs

Today:

- Learning r-juntas under uniform distribution.

Relevant Readings:

- Elchanan Mossel, Ryan O'Donnell, Rocco A. Servedio (2003). *Learning Juntas*
- T. Siegenthaler (1984). *Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Applications*
- Gregory Valiant (2012). *Finding Correlations in Subquadratic Time, with Applications to Learning Parities and Juntas with Noise*

2 Introduction

Definition 1. *Variable i in $f(x_1, ..., x_n)$ is relevant if $\exists\ x \in \{-1,1\}^n$ such that $f(x^{i\leftarrow 1}) \neq f(x^{i\leftarrow -1})$.*

Definition 2. *Function $f : \{-1,1\}^n \rightarrow \{-1,1\}$ is an r-junata if f has $k \leq r$ relevant variables.*

Example 3. *$x_{17} \oplus (x_{412} \vee (x_{916,774} \oplus x_{17}))$ is a 3-junta. This can be treated as a function of the 3 variables $x_{17}, x_{412}, x_{916,774}$.*

Observation 4. *Let f be an r-junta. Then f has a DT of size 2^r and a DNF of $s \leq 2^r$ terms.*

So to learn $\omega(1) - size$ DNFs, DTs in poly(n) time, we need to be able to learn juntas.

Let $\mathcal{C}_r = \{$all r-juntas over n variables$\}$. By learning r-juntas, we mean learn any r-junta over the n variables. So it's very natural to ask: What can we hope for with respect to learning $\mathcal{C}_r$?

3 Description Length

We first talk about the description length of an r-junta. The description length of an r-junta is $r \log n + 2^r$ bits. First, since there are n variables, we need $\log n$ bits to represent each variable. Then for r variables, we need $r \log n$ bits. Second, for r variables, the truth table contains 2^r entries. So it takes 2^r bits to write down the truth table. To sum up, the description length of an r-junta is $r \log n + 2^r$.

Note that all the r variables in an r-junta are relevant, this means that 2^r is the minimum number of examples we could hope for. So, the running time of the learning algorithm for r-juntas is associated with n and 2^r. It is unknown whether we can learn r-juntas in time poly$(n, 2^r)$.

4 Observations and Approaches

Observation 5.

$$|\mathcal{C}_r| \leq n^r \cdot 2^{2^r}$$

This observation follows by the fact that there are $\binom{n}{r}$ possible ways to choose r relevant variables from n variables and there are 2^{2^r} possible truth tables over r variables.

From this observation, we know that we can learn an r-junta in $O(n^r \cdot 2^{2^r})$ by trying every possible concept in $\mathcal{C}_r$. Due to the 2^{2^r} part, this is not an efficient algorithm.

Observation 6. *To learn r-juntas, it suffice to be able to find relevant variables. Given the r relevant variables, $O(r \cdot 2^r)$ examples suffice to fill in the truth table.*

We can get $r \cdot 2^r$ by applying the method of solving coupon collector's problem. Let T be the time to collect all k coupons. Here, T is the number of examples to collect all 2^r truth table entries. That is, $k = 2^r$. So we get

$$E(T) = k \ln k = 2^r \cdot \ln(2^r) = r \cdot 2^r$$

Now the main focus of learning r-juntas turns to how to find the r relevant variables efficiently.

Observation 7. *If variable i is irrelevant in f, then $Inf_i(f) = 0$ and $\hat{f}(S) = 0\ \forall S \ni i$. If variable i is relevant in r-junta f, then $Inf_i(f) \geq \frac{1}{2^r}$ and some $S \ni i$ must have $|\hat{f}(S)| \geq \frac{1}{2^r}$.*

The first part of the observation can be easily verified by the definition of influence and relevant. We now just prove "If i is relevant to f, then some $S \ni i$ must have $|\hat{f}(S)| \geq \frac{1}{2^r}$".

Recall from previous lectures, we have

$$Inf_i(f) = \sum_{S \ni i} \hat{f}(S)^2$$

Then for relevant variable i, if no $S \ni i$ has $|\hat{f}(S)| \geq \frac{1}{2^r}$, we get

$$\begin{aligned}
Inf_i(f) &= \sum_{S \ni i} \hat{f}(S)^2 \\
&\leq 2^r \cdot \max_{S \ni i} (\hat{f}(S)^2) \\
&< 2^r \cdot (\frac{1}{2^r})^2 \\
&= \frac{1}{2^r}
\end{aligned}$$

For the second inequality, note that from the first part of the observation, we know if S has irrelevant variables, then $\hat{f}(S) = 0$. So, there are at most 2^r $S_{S \ni i}$ whose $\hat{f}(S_{S \ni i}) \neq 0$.

This contradicts with the fact that $Inf_i(f) \geq \frac{1}{2^r}$. So some $S \ni i$ must have $|\hat{f}(S)| \geq \frac{1}{2^r}$.

From this observation, we know that we can learn concepts from $\mathcal{C}_r$ in $n^r \cdot poly(2^r)$. And here is the algorithm:

- For all S with $1 \leq |S| \leq r$, we estimate $\hat{f}(S)$ with accuracy of $\pm 0.1 \cdot \frac{1}{2^r}$. Whenever we find a S with $\hat{f}(S) \neq 0$, add all variables in S to the collection of relevant variables.
- After we get all relevant variables by this way, we can learn the r-junta with $O(r \cdot 2^r)$ more examples.

Note that there are n^r possible S here. And given a S, the estimation takes poly(2^r) time. This is due to the accuracy parameter $0.1 \cdot \frac{1}{2^r}$ So in total, this algorithm takes $n^r \cdot \text{poly}(2^r)$ time. Due to the n^r part, this algorithm is still not good enough.

5 Main Result

We can learn $\mathcal{C}_r$ in $n^{0.704 \cdot r} \cdot \text{poly}(2^r)$ time. This result is shown in the paper "Learning Juntas" by Elchanan Mossel, Ryan O'Donnell and Rocco Servedio in 2003.

Note that the state of the art result for learning r-juntas is $n^{0.61 \cdot r} \cdot \text{poly}(2^r)$ by Greg Valiant.

5.1 High level idea of the method

We will look at 2 different polynomial representations of f:

- Fourier representation
- GF(2) representation

The intuition of the method is if f is "bad" for Fourier-based learning, i.e. all its non-constant Fourier coefficients are on high-degree monomials, then f must be "good" for GF(2)-based learning.

5.2 Find 1 relevant variable efficiently is enough

From previous observations, we know that the most difficult part of learning juntas is how to find r relevant variables efficiently. We now show to learn r-juntas efficiently, all we need to do is to find 1 relevant variable efficiently.

Claim 8. *Suppose A is a T(n,r)-time algorithm which finds a relevant variable in an r-junta, given uniform (x, f(x)) random examples. Then there's an algorithm to learn r-juntas running in $T(n,r) \cdot \mathrm{poly}(n, 2^r)$ time.*

Proof. We will use A to get relevant variable "i", or find out no variable is relevant. If no variable is relevant, then done.

Now we talk about the situation that the function is not a constant function. Then, we will run A in a recursive style. It's like a binary search:

1. Run A to get a relevant variable i.
2. Run A on $(x, f(x))|_{x_i=1}$.
3. Run A on $(x, f(x))|_{x_i=-1}$.

In this way, we build a decision tree with depth $\leq r$. Note that every time we get a relevant variable i, we will split the examples into 2 roughly equal-sized subsets. Since the subroutine to find a relevant variable requires certain number of examples, the more relevant variables we want to find, the more examples we must have. So, this algorithm has a 2^d slow factor slow down at depth d. The decision tree has a depth $\leq r$, so the slow down factor is at most 2^r. We will run A at most $2^{r+1}-1$ times. Also, note that we have no restrictions on $T(n,r)$ and all examples are from $\{-1,1\}^n$. So we will add restriction $\mathrm{poly}(n)$ in the running time. At last, we get the running time

$$\begin{aligned} T(r) &\leq (2^{r+1}-1) \cdot 2^r \cdot T(n,r) \cdot \mathrm{poly}(n) \\ &= T(n,r) \cdot \mathrm{poly}(n, 2^r) \end{aligned}$$

The running time of the algorithm to learn r-juntas is $T(n,r) \cdot \mathrm{poly}(n, 2^r)$.

□

5.3 Fourier-based learning

Fact 9. *If f has $\hat{f}(S) \neq 0$ for some S with $1 \leq |S| \leq c \cdot r$, then we can find a relevant variable in $\leq 2^r \cdot n^{cr}$ time.*

From observation 7, we know that if $\hat{f}(S) \neq 0$, then all variables in S are relevant. To find a relevant variable, we just need to find a S with $\hat{f}(S) \neq 0$. To find such a S, we simply try every possible S with $1 \leq |S| \leq c \cdot r$. There are $n^{c \cdot r}$ different S in total. And for the $\leq r$ relevant variables, there are 2^r possible cases. Combine the two together, we get $\leq 2^r \cdot n^{cr}$.

Later, we'll see if f has no such S as stated in this fact, then a GF(2)-based learning algorithm will work.

5.4 GF(2)-based learning

GF(2) is the Galois field of two elements. It is the smallest finite field. The GF(2) representation of f is a multilinear polynomial over the field $GF(2) = \{0, 1\}$, and all math is done with modulo 2. This ensures that the field is closed under addition and multiplication. In GF(2), 0 is equivalent to false, 1 is equivalent to true, addition is equivalent to parity function, and multiplication is equivalent to AND function.

Fact 10. *Let $f : \{0, 1\}^n \to \{0, 1\}$. Then f has*

1. *a unique representation as a multilinear polynomial P_R over real numbers. All coefficients are integers.*
2. *a unique representation as a GF(2) polynomial $P_{GF(2)}$.*
3. *If p_1, p_2 are 2 degree-d GF(2) polynomials for f and g, and $f \neq g$, then $Pr_x[f(x) \neq g(x)] \geq \frac{1}{2^d}$.*

The proof is left as the official homework.

An easy relation between P_R and $P_{GF(2)}$: we can get $P_{GF(2)}$ from P_R by reducing all coefficients mod 2.

$$P_R = \sum_{S \subseteq [n]} C_S X_S \Rightarrow P_{GF(2)} = \sum_{S \subseteq [n]} (C_s \mod 2) \chi_S$$

So $deg(P_{GF(2)}) \leq deg(P_R)$ for all f.

Example 11. *Any parity function has a deg-1 GF(2) polynomial:* $x_3 + x_6 + x_7 + x_{10}$.

Note that any parity function's R-polynomial is deg-n.

Example 12. $AND(x_1, ..., x_n)$ *has deg-n GF(2) polynomial:* $x_1 x_2 x_3 ... x_n$.

As mentioned, in $GF(2)$, addition is parity function and multiplication is AND function. So, a GF(2) polynomial is a parity of ANDs. So, the main focus of GF(2) learning is whether we can learn parity functions over some unknown subset of variables easily by "thinking GF(2)".

Claim 13. *Let* $C = \{$*all* 2^n *PARs over* $x_1, ..., x_n\}$. *There's an algorithm to PAC learn* C *under any distribution* $\mathcal{D}$ *over* $\{0,1\}^n$ *in time* $(\frac{n}{\epsilon} \cdot \log \frac{1}{\delta})^w$, *where* $w \approx 2.374$ *is matrix multiplication exponent.*

Proof. First we talk about the matrix multiplication problem. Trivially, For two $n \times n$ matrices, the algorithm for matrix multiplication problem takes $O(n^3)$ time. That is, the matrix multiplication exponent w is 3. We know that we can solve this problem with $w \leq 2.374$.

We know from "Occam's Razor" that with probability $\geq 1 - \delta$, any PAR that is consistent with $O(\frac{n}{\epsilon} \cdot \log \frac{1}{\delta})$ examples is ϵ-accurate. So, we can learn a parity function using $O(\frac{n}{\epsilon} \cdot \log \frac{1}{\delta})$ examples.

Now, we will use $m = O(\frac{n}{\epsilon} \cdot \log \frac{1}{\delta})$ examples to build m linear equations. And the GF(2) PAR learning problem becomes a problem of solving a system of m equations: $Ma = b$. M is a $m \times n$ matrix, each row of M represents one example's input $x_1, x_2, ..., x_n$. a is a $n \times 1$ vector that each row represents if every variable i is in the parity function. b is a $n \times 1$ vector represents all examples' outputs.

To sum up, from "Occam's Razor" we know that $m = O(\frac{n}{\epsilon} \cdot \log \frac{1}{\delta})$ examples is enough to give a ϵ-accurate parity function. From the first part of the proof, we know that we can solve it in $O((\frac{n}{\epsilon} \cdot \log \frac{1}{\delta})^w)$ time, where $w \leq 2.374$. $\square$

Using the same approach, we can get the following claim.

Claim 14. *Let* $C = \{$*all deg-d GF(2) polynomials over* $x_1, ..., x_n\}$. *We can PAC learn any deg-d GF(2) polynomial to accuracy* ϵ *under any distribution* $\mathcal{D}$ *in time* $O(\frac{n^d}{\epsilon} \cdot \log \frac{1}{\delta}) \approx \frac{n^{dw}}{\epsilon^w}$.

Note that we will learn r-juntas under uniform distribution on $\{0,1\}^n$. Since if the hypothesis given by this claim is not the target concept, then $error(h) \geq \frac{1}{2^d}$. In this case, we can set $\epsilon = \frac{1}{2^{d+1}} < \frac{1}{2^d}$, then any ϵ-accurate hypothesis must be exactly the target concept we want to learn.

To sum up, we get the bottom line that we can exactly learn any deg-d GF(2) polynomial with this algorithm under uniform distribution in time $n^{w \cdot d} \cdot \text{poly}(2^d)$.

5.5 The Algorithm to find a relevant variable

Given r, c ($c < 1$; the reason will be given later), there is a algorithm to find a relevant variable:

1. For $d = 1, ..., c \cdot r$, estimate all $\hat{f}(S)$ with $|S| = d$ to $\pm\frac{0.1}{2^r}$. If we find one which is non-zero, stop and output any variable in S. If f is a constant function, then stop.
2. Otherwise, have $\hat{f}(S) = 0 \ \forall \ 1 \leq |S| \leq c \cdot r$. Then run algorithm to learn $(\alpha \cdot r) - deg$ GF(2) polynomials, using $\epsilon = \frac{1}{2^{r+1}}$.

Claim 15. *For $c = \frac{w}{w+1}$, $\alpha = \frac{1}{w+1}$, this algorithm works. And it runs in time $\approx O(n^{0.704 \cdot r})$.*

The running time can be easily verified. With the specific c and α, step 1 of the algorithm runs in $O(n^{c \cdot r}) = O(n^{\frac{w}{w+1} \cdot r})$. Step 2 of the algorithm runs in $O(n^{w \cdot d}) = O(n^{w \cdot \alpha \cdot r}) = O(n^{\frac{w}{w+1} \cdot r})$. So this algorithm runs in $O(n^{\frac{w}{w+1} \cdot r}) \approx n^{0.704 \cdot r}$.

Now we will prove this algorithm works. The high level idea of this proof is that if $\hat{f}(S) = 0 \ \forall 1 \leq |S| \leq c \cdot r$, then f is a deg-d GF(2) polynomial. In this proof, we will view f as $f : \{-1,1\}^r \to \{-1,1\}$.

We first give an useful definition.

Definition 16. *Let $f : \{-1,1\}^r \to \{-1,1\}$. We say f is d^{th} order correlation immune (d-c.i.) if $\hat{f}(S) = 0 \ \forall 1 \leq |S| \leq d$.*

The proof of the algorithm works will be done if we can show the following theorem is correct.

Theorem 17. *Let $f : \{-1,1\}^r \to \{-1,1\}$, $f \neq PAR(x_1, ..., x_r)$ and $f \neq -PAR(x_1, ..., x_r)$. Suppose f is d-c.i., then f has GF(2) polynomial of deg $\leq r - d$.*

Proof. First, suppose $d = r$. Since f is r-c.i., then f is a constant function and it has deg-0 GF(2) polynomials.

Now we assume $d < r$, we have

$$f(x) = \hat{f}(\emptyset) + \sum_{d<|s|\leq r} \hat{f}(S)\chi_s$$

Let $h(x) = f(x) \oplus PAR(x_1, ..., x_r)$, i.e. $h(x) = f(x) \cdot x_1 x_2 ... x_r = f(x) \cdot x_{[r]}$. The Fourier representation of h is:

$$\begin{aligned} h(x) &= \hat{f}(\emptyset)\chi_{[r]} + \sum_T \hat{f}(T)\chi_T \cdot \chi_{[r]} \\ &= \hat{f}(\emptyset)\chi_{[r]} + \sum_{0<|T|\leq d} \hat{f}(T)\chi_T \cdot \chi_{[r]} + \sum_{d<|T|\leq r} \hat{f}(T)\chi_T \cdot \chi_{[r]} \\ &= \hat{f}(\emptyset)\chi_{[r]} + 0 + \sum_{d<|T|\leq r} \hat{f}(T)\chi_T \cdot \chi_{[r]} \\ &= \hat{f}(\emptyset)\chi_{[r]} + \sum_{0\leq|T|<r-d} \hat{f}([r]\backslash T)\chi_T \end{aligned}$$

Case 1: $\hat{f}(\emptyset) = 0$. Then

$$h(x) = \sum_{0\leq|T|<r-d} \hat{f}([r]\backslash T)\chi_T$$

Let h' be

$$h'(y_1, ..., y_r) = \frac{1}{2} - \frac{h(1-2y_1, ..., 1-2y_r)}{2}$$

As we can see, h' is equivalent to h but with 0/1 inputs and outputs. Since $deg_R(h') < r - d$. So $deg_{GF(2)}(h') < r - d$.

Case 2: $\hat{f}(\emptyset) = 0$. As before,

$$\begin{aligned} h(x) &= \hat{f}(\emptyset)\chi_{[r]} + \sum_{0\le|T|<r-d} \hat{f}([r]\backslash T)\chi_T \\ h'(y_1,...,y_r) &= \frac{1}{2} - \frac{h(1-2y_1,...,1-2y_r)}{2} \end{aligned}$$

We will show that the contribution from $\hat{f}(\emptyset)\chi_{[r]}$ to GF(2) polynomial for h' doesn't give us $deg > r-d$. This whole contribution of this term is

$$\frac{-\hat{f}(\emptyset)}{2}\prod_{i=1}^{r}(1-2y_i) = \frac{-\hat{f}(\emptyset)}{2}\sum_{S\subset[r]}(-2)^{|S|}y_S$$

Fix $S' = \{1,2,...,r-d\}$. What's the coefficient of $y_{S'}$ in h'? In the $non-\emptyset$ term, it must be 0. And in the $\emptyset$ term, it's

$$\frac{-\hat{f}(\emptyset)}{2}(-2)^{r-d}$$

Previous fact tells us that this term must be an integer. Now consider any set of S of size $> r-d$. h''s coefficient on y_S is:

$$\frac{-\hat{f}(\emptyset)}{2}(-2)^{|S|} = \frac{-\hat{f}(\emptyset)}{2}(-2)^{r-d}\cdot 2^t$$

That is, this term is an even integer. So $h'_{GF(2)}$ has coefficient 0 on every S of size $> r-d$. So $deg(h'_{GF(2)}) \le r-d$. $\square$