0368.4163 Randomness and Computation May 27, 2009

Lecture 10

Lecturer: Ronitt Rubinfeld *Scribe: Elya Dolev and Gilad Tsur*

Today:

1. Learning under the uniform distribution
2. Learning via Fourier representation.

1 Uniform Distribution Learning

Definition 1 *Given hypothesis h,* error of h with respect to f *is* $\text{error}(h) = \Pr_{x \in_U}[f(x) \neq h(x)]$*, where U is the uniform distribution.*

Definition 2 *A* uniform distribution learning algorithm *for concept class $\mathcal{C}$ is an algorithm $\mathcal{A}$ such that*

1. *$\mathcal{A}$ is given $\epsilon, \delta > 0$, and has access to a set of samples $(x, f(x))$, where each x is chosen uniformly from the domain.*
2. *$\mathcal{A}$ outputs h such that with probability at least $\geq 1 - \delta$, error of h with respect to f is at most ϵ.*

Note that if f is arbitrary, then the learning task is impossible without seeing almost all the inputs of f, however, if we know that f is in a certain class of functions, referred to as the concept class $\mathcal{C}$, then the learning task may become possible. We are given access to an example oracle, and to the parameters δ and ϵ. The value δ is often referred to as the "security" or "confidence parameter", and ϵ is the approximation error or "accuracy parameter".

1.1 Parameters of Interest

- m: "Sample complexity". The number of randomly chosen examples that we get from the oracle.
- ϵ: Accuracy parameter. Discussed above.
- δ: Security parameter. Discussed above.
- Running time? Might be different than the sample complexity or query complexity. Note that, trivially, $m \leq$ running time. We would like to get something polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$ (actually, $\log \frac{1}{\delta}$) $\log |\mathcal{C}|$, $\log |D|$. Note that the requirement on the domain space assumes that we fully read the values $x, f(x)$ given to us from the example oracle, so they should be simple to describe.
- Description of h should be relatively compact. Some models restrict this further.

Note that h is not necessarily in $\mathcal{C}$. In some models, "proper learning algorithms," this is the case, but we will not restrict our attention to them.

1.2 Remarks

- This Uniform Distribution Learning model is a special case of the PAC (probably approximately correct) model. More generally, $\text{Ex}(f)$ for unknown distribution $\mathcal{D}$.

$$\text{error}(h) = \Pr_{x \in \mathcal{D}}[f(x) \neq h(x)]$$

- Can get the dependence on δ to be $\log \frac{1}{\delta}$.

2 Brute Force Algorithm

If the running time doesn't matter, only sample complexity, then it is easy to write learning algorithms, but as we will see, the algorithm that makes this possible is infeasible in practice.

- Draw $M = \frac{1}{\epsilon}(\ln|\mathcal{C}| + \ln\frac{1}{\delta})$ samples. $\mathcal{C}$ is generally regarded as a class of exponential size.
- Search over all $h \in \mathcal{C}$ until one h labels all examples correctly and output it. Given multiple choices, chose arbitrarily. Notice that the sample complexity is fine, because we reuse our samples for each iteration of the search.

Motivation: h is bad, if the error of h with respect to $f \geq \epsilon$. Whether $h = f$ on all inputs or just most, doesn't really matter.

$$\begin{aligned} Pr[\text{specific bad } h \text{ is consistent with examples}] &\leq (1-\epsilon)^M \\ Pr[\text{there exist a bad } h \text{ that "survives"}] &\leq |\mathcal{C}|(1-\epsilon)^M \\ &\leq |\mathcal{C}|(1-\epsilon)^{\frac{1}{\epsilon}(\ln|\mathcal{C}|+\ln\frac{1}{\delta})} \\ &\leq \delta \end{aligned}$$

Therefore, this algorithm is unlikely to output any bad h.

Remark Ignoring the sample size bounds leads to bad statistics, where the sample size is insufficient for an inordinately large class space to rule out all of the bad h possibilities. The key is to define in advance the concept class, rather than expanding the concept class to fit "interesting" results.

3 Example: Monomial Functions

- $\mathcal{C}$ = conjunctions over $\{0,1\}^n$ (including negation). $|\mathcal{C}| = 3^n$
- Example: $f = x_i x_j \bar{x_k}$
- Cannot learn efficiently with 0 error. Think about a very long conjunction, which is 0 unless none of the variables are 0. In a polynomial number of queries, there is no way to detect it is not the all zeros function.
- Brute Force needs $\frac{1}{\epsilon}\left(\ln 3^n + \ln\frac{1}{\delta}\right) = O\left(\frac{n}{\epsilon} + \frac{1}{\epsilon}\ln\frac{1}{\delta}\right)$ samples.

3.1 Poly-time Algorithm

We say that a sample (or an input) $(x, f(x))$ is *positive* if $f(x) = 1$. If $f(x) = 0$, we say that the sample is *negative*. Let us describe the algorithm. We will set k that appears in the algorithm later on.

1. Draw $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ samples to estimate the fraction of positive inputs within $\epsilon/4$ with probability at least $1 - \frac{\delta}{2}$.
2. If in less than $\epsilon/2$ fraction of samples are positive, output $h(x) = 0$.
3. Take samples until have at least k positive samples.
4. Let $V = \{i : x_i \text{ is the same in all positive examples}\}$.

5. Output hypothesis $h(x) = \bigwedge_{i \in V} x_i^b$, where b is the complement or the variable.

Why does this work? First we check if the function is sufficiently far from the all-zero function. If it is not, we can output the all-zero function as our hypothesis (Step 2). Otherwise, we know in Steps 3–5 that $\Pr_x[f(x) = 1] \geq \epsilon/4$.
If x_i is in f, then it must have the same value for every positive sample, and $x_i \in V$.
If x_i is not in f, $\Pr[x_i \in V] \leq \Pr[x_i$ is set the same in each positive example$] \leq \frac{1}{2^{k-1}}$. Thus using union bound, $\Pr[$any x_i survives in $V] \leq \frac{n}{2^{k-1}}$, and we want this to be at most δ. Thus we need $k = O(\log(\frac{n}{\delta}))$ positive examples, which we will draw with sufficiently high probability if we will draw a total number of $O(\frac{1}{\epsilon}\log(\frac{n}{\delta})\frac{1}{\delta})$ samples.

Notice that learning requires logarithmically many queries, even for singletons, while dictator testing takes a constant number of queries. So, testing can be much faster than learning.

4 Learning via Fourier Coefficients

Recall:

- The Fourier representation of f is: $f(x) = \sum_z \hat{f}(x)\chi_z(x)$.
- The Fourier coefficient for s can be computed by: $\hat{f}(s) = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x)\chi_s(x)$.
- The distance of f from the linear function chi_s can be expressed in terms of its Fourier coefficient corresponding to s as follows: $dist(f, \chi_s) = \frac{1}{2}(1 - \hat{f}(s))$.

Approximating a single Fourier coefficient of an unknown function:

Lemma 3 *We can approximate any specific Fourier coefficient s to within γ (i.e. $\left|output - \hat{f}(s)\right| < \gamma$) with probability $\geq 1 - \delta$ in $O\left(\frac{1}{\gamma^2}\log\frac{1}{\delta}\right)$ samples.*

Proof We have $\hat{f}(s) = 2 \times \Pr[f = \chi_s] - 1$, and we can estimate $\Pr[f = \chi_s]$ to within $\pm\gamma/2$ using Chernoff bounds. ■

It is thought to be unlikely in this model that one can efficiently find the "heavy" Fourier coefficients with non-trivial values. Exhaustive search is intractable, and no other method is immediately clear. If we are dealing with a function class where most of the weight is in the coefficients with small s, we can exhaustively check these values. First, however, we will describe and analyze the *Low Degree Algorithm*.

5 Preliminaries

We need a few definitions.

Definition 4 *A function $f : \{\pm 1\}^n \to \mathbb{R}$ is said to have $\alpha(\epsilon, n)$ concentration if*

$$\sum_{S \subseteq [n], |S| > \alpha(\epsilon,n)} \hat{f}(S)^2 \leq \epsilon, \ \forall\ 0 < \epsilon < 1. \tag{1}$$

In particular, for Boolean functions $f : \{\pm 1\}^n \to \{\pm 1\}$, the above relation is equivalent to

$$\sum_{S \subseteq [n], |S| \leq \alpha(\epsilon,n)} \hat{f}(S)^2 \geq 1 - \epsilon, \ \forall\ 0 < \epsilon < 1. \tag{2}$$

This follows from Parseval's theorem, which states that

$$\sum_{S \subseteq [n]} \hat{f}(S)^2 = 1. \tag{3}$$

Intuitively, if α assumes small values, then the concentration of the function is on the low-degree terms of its Fourier decomposition. (Throughout these notes, we will use the terms *Fourier concentration* and *Fourier degree* interchangeably.)

Let us now consider some functions which satisfy this property, i.e. have low Fourier concentration. The first example we consider is the class of *junta* functions. These functions depend on only $k \ll n$ variables. Clearly, if f is such a function which does not depend on a variable i, then

$$\hat{f}(S) = 0, \ \forall \ S \subseteq [n], \ i \in S. \tag{4}$$

Since the junta function depends on only k variables (without loss of generality, assume that these variables are $1 \ldots k$), a direct consequence of the above observation is that

$$\sum_{S \subseteq [n], |S| > k} \hat{f}(S)^2 = 0. \tag{5}$$

Thus, the junta function has Fourier degree at most k for any value of ϵ.

Our next example is the following slightly unusually defined AND function.

$$\mathbf{and}(x_1, x_2, \ldots, x_k) = \begin{cases} 1 & \text{if } \forall i, \ x_i = -1 \\ -1 & \text{otherwise.} \end{cases} \tag{6}$$

Theorem 5 *The* **and** *function defined above has Fourier concentration* $\log(4/\epsilon)$.

Proof We consider two cases depending on the value of ϵ.
Case 1: $k \leq \log(4/\epsilon)$. In this case, from our analysis of the junta function above, we directly obtain

$$\sum_{S \subseteq [n], |S| > \log(4/\epsilon)} \hat{f}(S)^2 = 0 < \epsilon. \tag{7}$$

Case 2: $k > \log(4/\epsilon)$. Define

$$g(x) = \begin{cases} 1 & \text{if } \forall i, \ x_i = -1 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$g(x) = (\frac{1 - x_1}{2})(\frac{1 - x_2}{2}) \ldots (\frac{1 - x_k}{2}) \tag{9}$$

$$= \sum_{S \subseteq [k]} \frac{(-1)^{|S|}}{2^k} \chi_S. \tag{10}$$

Now,

$$\mathbf{and}(x) = 2g(x) - 1 \tag{11}$$

$$= (-1 + 2^{1-k}) + \sum_{S \subseteq [k], |S| > 0} (-1)^{|S|} 2^{1-k} \chi_S. \tag{12}$$

Clearly, there are at most 2^k non-zero Fourier co-efficients corresponding to characters S such that $|S| > 0$. Thus,

$$\sum_{S \subseteq [k], |S| > \log(4/\epsilon)} \hat{f}(S)^2 \leq \sum_{S \subseteq [k], |S| > 0} \hat{f}(S)^2 \tag{13}$$

$$\leq [(-1)^{|S|} 2^{1-k}]^2 2^k \tag{14}$$

$$= \frac{4}{2^k} \tag{15}$$

$$< \epsilon. \tag{16}$$

Input: Oracle access to function f under uniform sampling, Fourier degree of f denoted by $d = \alpha(\epsilon, n)$, accuracy parameter τ and confidence or security parameter δ.

1. Let $m = O(\frac{n^d}{\tau} \ln \frac{n^d}{\delta})$.
2. Collect m samples from the function.
3. For each set $S \subseteq [n]$ such that $|S| \leq d$, let C_S be an estimate for $\hat{f}(S)$.
4. Let $h = \sum_{S \subseteq [n], |S| \leq d} C_S \chi_S$.
5. Output sign(h) as the estimate of f.

Figure 1: The low-degree algorithm

The last inequality follows from the assumption that $k > \log(4/\epsilon)$. ■

6 The Low Degree Algorithm

The low degree algorithm is a uniform distribution learning algorithm which is efficient for functions with low Fourier degree. The algorithm is presented in Figure 1. As an optimization, note that the result of a single set of m queries to the function f can be re-used in estimating C_S for each S, since we do not use the independence of the estimates in our analysis of correctness.

We now analyze the low-degree algorithm. We prove two lemmas which will lead us to the ultimate result.

Lemma 6 *If f has $\alpha(\epsilon, n)$ Fourier concentration, then h computed by the algorithm satisfies*

$$E_x[(f(x) - h(x))^2] \leq \epsilon + \tau, \tag{17}$$

with probability $\geq 1 - \delta$.

To prove this lemma, we need the following lemma.

Lemma 7 *With probability $\geq 1 - \delta$, for each S satisfying $|S| \leq d$,*

$$|C_S - \hat{f}(S)| \leq \gamma, \tag{18}$$

where $\gamma = \sqrt{\frac{\tau}{n^d}}$.

Proof Since the $O(\frac{n^d}{\tau} \log \frac{n^d}{\delta}) = O(\frac{1}{\gamma^2} \log \frac{n^d}{\delta})$ samples used to estimate any Fourier coefficient $\hat{f}(S)$ are independent, we can claim, using Chernoff bound, that

$$|C_S - \hat{f}(S)| > \gamma \tag{19}$$

with probability $\leq \frac{\delta}{n^d}$. Since there are $\leq n^d$ sets S such that $|S| \leq d$, the lemma follows using union bound. ■

We now use this lemma to prove Lemma 6.

Proof of Lemma 6 Define

$$g(x) = f(x) - h(x). \tag{20}$$

Since the Fourier decomposition is linear, for each $S \subseteq [n]$,

$$\hat{g}(S) = \hat{f}(S) - \hat{h}(S). \tag{21}$$

From the algorithm, for any S such that $|S| > d$, $\hat{h}(S) = 0$. Thus,

$$\begin{aligned} \sum_{S\subseteq[n],|S|>d} \hat{g}(S)^2 &= \sum_{S\subseteq[n],|S|>d} \hat{f}(S)^2 & (22)\\ &\leq \epsilon. & (23)\end{aligned}$$

The second inequality holds since $d = \alpha(\epsilon, n)$.

On the other hand, for each S such that $|S| \leq d$,

$$\begin{aligned} \hat{g}(S)^2 &= (\hat{f}(S) - \hat{h}(S))^2 & (24)\\ &\leq \gamma^2, & (25)\end{aligned}$$

with probability $\geq 1 - \delta$. The second inequality follows from Lemma 6. Since there are $\leq n^d$ sets S such that $|S| \leq d$,

$$\begin{aligned} \sum_{S\subseteq[n],|S|\leq d} \hat{g}(S)^2 &\leq \gamma^2 n^d & (26)\\ &= \tau. & (27)\end{aligned}$$

We now have

$$\begin{aligned} E_x[(f(x) - h(x))^2] &= E[g(x)^2] & (28)\\ &= \sum_{S\subseteq[n]} \hat{g}(S)^2 & (29)\\ &= \sum_{S\subseteq[n],|S|\leq d} \hat{g}(S)^2 + \sum_{S\subseteq[n],|S|>d} \hat{g}(S)^2 & (30)\\ &\leq \tau + \epsilon, & (31)\end{aligned}$$

with probability $\geq 1 - \delta$. The second equation follows from Plancherel's theorem. ■

We now prove another lemma which will be useful in the final result.

Lemma 8 *For any functions $f : \{\pm 1\}^n \to \{\pm 1\}$ and $h : \{\pm 1\}^n \to \mathbb{R}$,*

$$Pr[f(x) \neq \text{sign}(h(x))] \leq E_x[(f(x) - h(x))^2]. \tag{32}$$

Proof We have

$$E_x[(f(x) - h(x))^2] = \frac{1}{2^n} \sum_x (f(x) - h(x))^2, \tag{33}$$

while

$$Pr[f(x) \neq \text{sign}(h(x))] = \frac{1}{2^n} \sum_x \mathbf{1}(x), \tag{34}$$

where $\mathbf{1}$ is defined as

$$\mathbf{1}(x) = \begin{cases} 1 & \text{if } f(x) \neq \text{sign}(h(x)) \\ 0 & \text{otherwise.} \end{cases} \tag{35}$$

Now, for any x, if $f(x) \neq \text{sign}(h(x))$, then

$$(f(x) - h(x))^2 \geq 1 = \mathbf{1}(x), \tag{36}$$

and if $f(x) = \text{sign}(h(x))$, then

$$(f(x) - h(x))^2 \geq 0 = \mathbf{1}(x). \tag{37}$$

Thus,

$$\sum_x (f(x) - h(x))^2 \geq \sum_x \mathbf{1}(x). \tag{38}$$

■

We now arrive at our Main Theorem. However, before stating it, we need another definition.

Definition 9 *The Fourier concentration of a concept class of functions is the maximum Fourier concentration among all the functions in the class.*

Theorem 10 (Main Theorem) *Let concept class $\mathcal{C}$ have Fourier concentration $d = \alpha(\epsilon, n)$. Then, there exists a $q = O(\frac{n^d}{\epsilon} \log \frac{n^d}{\delta})$-sample uniform distribution learning algorithm for $\mathcal{C}$ (i.e. the algorithm uses q uniformly distributed independent samples and with probability $\geq 1 - \delta$, outputs a hypothesis h' such that $Pr[f(x) \neq h'(x)] \leq 2\epsilon$).*

Proof The algorithm is to simply run the low-degree algorithm for $\tau = \epsilon$. The error bound follows from Lemmas 6 and 8 given above. ■

What class of functions can we learn using the low-degree algorithm? One such large class of functions that can be learned in quasi-polynomial ($O(n^{\text{polylog}n})$) time using this algorithm are those which can be represented by constant depth circuits. Recall that a circuit consists of gates (typically AND, OR and NOT gates; we assume that the first two types of gates have unbounded fan-in), constants 0 and 1 and variables $X_1, X_2, \ldots, X_n$. It turns out that any function can be represented by a constant depth circuit, but we would also like to ensure that the size of the circuit is polynomial in the number of input variables n. So, one central question is whether all functions can be computed using constant-depth, polynomial-sized circuits? The answer is no, via a counting argument. In fact, it was proved by Furst, Saxe and Sipser that even the class of parity functions cannot be computed using constant depth, polynomial sized circuits.

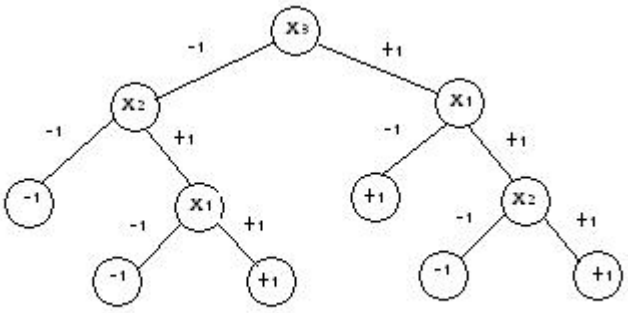
But what about functions which are computable using constant depth, polynomial sized circuits? The following theorem explores the Fourier concentration of such functions.

Theorem 11 (Håstad, Linial Mansour Nisan) *For any function f computable using a size-s depth-d circuit,*

$$\sum_{S \in [n], |S| > t} \hat{f}(S)^2 \leq \alpha, \tag{39}$$

for $t = O((\log \frac{2s}{\alpha})^{d-1})$.

Taking s as a polynomial in n, d as a constant and $\alpha = O(\epsilon)$ gives us $t = O(\log^d \frac{n}{\epsilon})$. Using our main theorem, we can conclude that the number of samples required for learning such a constant depth polynomial sized circuit is $n^{O(\log^d \frac{n}{\epsilon})}$, which is quasipolynomial in n. For DNF formulas, this bound can be improved to $O(n^{O(\log \log n)})$ samples.



Another family of functions that can be learned using the low degree algorithm is that of functions computable by decision trees with few nodes. To characterize these functions we first define the function that represents a single *decision path* ℓ in the decision tree: $f_\ell(x) = \prod_{i \in V_\ell} \frac{1 \pm x_i}{2}$, where V_ℓ is the set of variables visited on path ℓ. The plus or minus sign corresponds to the path's descent left or right in the decision tree. The Fourier representation for the path function is $\frac{1}{2^{|V_\ell|}} \sum_{S \subseteq |V_\ell|} (\pm 1)\chi_S$. The function computed by the whole decision tree can thus be written as $f(x) = \sum_\ell f_l(x) val(\ell)$ where $val(\ell)$ is the value at the leaf reached by ℓ. Note that exactly one value $f_\ell(x)$ will equal 1 for any value x.

Finally, an additional class of families that can be learnt is that of halfspaces and combinations of half spaces. We define a halfspace function as follows: $h : \pm 1^n \rightarrow \pm 1$ be such that $h(x) = sign(w \cdot x - \theta)$ for some w and θ.

Theorem 12 *h has Fourier concentration $441/\epsilon^2$*

Corollary 13 *The low degree algorithm learns halfspaces under the uniform distribution with $n^{O(1/\epsilon^2)}$ queries*

Note that other algorithms require only n^5 queries, so this isn't so great. However, any function g which is a function of k halfspaces has Fourier concentration $O(k^2/\epsilon^2)$ and as a corollary the low degree algorithm learns functions of k halfspace with $n^{O(k^2/\epsilon^2)}$ queries, which is better than what was known before.