6.842 Randomness and Computation 04/04, 2012

Lecture 15

Lecturer: Ronitt Rubinfeld *Scribe: Mohsen Ghaffari*

1 Recap

Definition 1 *Consider function $f : \{\pm 1\}^n \to R$. We say f has $\alpha(\epsilon, n)$-Fourier Concentration if*

$$\sum_{S \subseteq [n]\,,\, |S| > \alpha(\epsilon,n)} \hat{f}^2(S) \leq \epsilon$$

Theorem 2 *If $\mathcal{C}$ has Fourier Concentration at most $d = \alpha(\epsilon, n)$, then there is an $O(\frac{n^d}{\epsilon})$ sample-uniform learning algorithm for $\mathcal{C}$.*

Definition 3 *Let $N_\epsilon(x)$ be what we get from randomly flipping each bit of x with probability ϵ (iid). Then noise sensitivity of function f, $ns_\epsilon(f)$, is defined as $ns_\epsilon(f) = \Pr_{x\in\{\pm1\}^n}\big[f(x) \neq f(N_\epsilon(x))\big]$.*

Theorem 4 *For any linear threshold function f, we have $ns_\epsilon(f) \leq 8.8\sqrt{\epsilon}$.*

2 Today

2.1 Noise Sensitity vs. Fourier Concentration

Theorem 5 *Consider function $f : \{\pm 1\}^n \to \{\pm 1\}$. Then, we have*

$$ns_\epsilon(f) = \frac{1}{2} - \frac{1}{2}\sum_{S\subseteq[n]} (1-2\epsilon)^{|S|}\hat{f}^2(S)$$

Proof By Definition 3, we have

$$\begin{aligned}
ns_\epsilon(f) &= \Pr_{x\in\{\pm1\}^n}\big[f(x) \neq f(N_\epsilon(x))\big] \\
&= \Pr_{x\in\{\pm1\}^n, y = N_\epsilon(x)}\big[f(x) \neq f(y)\big] \\
&= \mathbb{E}_{x,y}\big[\mathbf{1}_{f(x)\neq f(y)}\big] \\
&= \mathbb{E}_{x,y}\big[\frac{(f(x)-f(y))^2}{4}\big] \\
&= \mathbb{E}_{x,y}\big[\frac{2-2f(x)f(y)}{4}\big] \\
&= \frac{1}{2} - \frac{1}{2}\mathbb{E}_{x,y}\big[f(x)f(y)\big] \\
&= \frac{1}{2} - \frac{1}{2}\sum_{s,T\subseteq[n]} \hat{f}(S)\hat{f}(T)\,\mathbb{E}_{x,y}\big[\chi_S(x)\chi_T(y)\big] \\
&= \frac{1}{2} - \frac{1}{2}\sum_{s,T\subseteq[n]} \hat{f}^2(S)\,\mathbb{E}_{x,y}\big[\chi_S(x)\chi_S(y)\big] \qquad (1)
\end{aligned}$$

Now, we evaluate $\mathbb{E}_{x,y}\big[\chi_S(x)\chi_S(y)\big]$. Let e_{x_i} (resp. e_{y_i}) be the unit vector that has value x_i (resp y_i) at position i and 1 at all the other places. Then, we have

$$\mathbb{E}_{x,y}\big[\chi_S(x)\chi_S(y)\big] = \mathbb{E}_{x,y}\big[\prod_{i=1}^{n}\chi_S(e_{x_i})\chi_S(e_{y_i})\big]$$

$$
\begin{aligned}
&= \mathbb{E}_{x,y}\big[\prod_{i\in S}\chi_S(e_{x_i})\chi_S(e_{y_i})\big] \\
&= \prod_{i\in S}\mathbb{E}_{x,y}\big[\chi_S(e_{x_i})\chi_S(e_{y_i})\big] \\
&= \prod_{i\in S}(\Pr[\chi_S(e_{x_i}) = \chi_S(e_{y_i})] - \Pr[\chi_S(e_{x_i}) \neq \chi_S(e_{y_i})]) \\
&= \prod_{i\in S}(\Pr[x_i = y_i] - \Pr[x_i \neq y_i]) \\
&= \prod_{i\in S}(1-2\epsilon) \\
&= (1-2\epsilon)^{|S|} \qquad (2)
\end{aligned}
$$

Hence, substituting this into (1), we get

$$ns_\epsilon(f) = \frac{1}{2} - \frac{1}{2}\sum_{S\subseteq[n]}(1-2\epsilon)^{|S|}\hat{f}^2(S)$$

■

Using this result, we can get the following connection between noise sensitivity of a function and its Fourier Concentration.

Theorem 6 *Consider function $f : \{\pm1\}^n \to \{\pm1\}$. Then, for any $0 < \gamma < \frac{1}{2}$, we have*

$$\sum_{S\subseteq[n],|S|\geq\frac{1}{\gamma}}\hat{f}^2(S) \leq 2.32\ ns_\gamma(f)$$

Proof Using Theorem 5, we have

$$
\begin{aligned}
2\ ns_\gamma(f) &= 1 - \sum_{S\subseteq[n]}(1-2\gamma)^{|S|}\hat{f}^2(S) \\
&= \sum_{S\subseteq[n]}\hat{f}^2(S) - \sum_{S\subseteq[n]}(1-2\gamma)^{|S|}\hat{f}^2(S) \\
&= \sum_{S\subseteq[n]}\big(1-(1-2\gamma)^{|S|}\big)\ \hat{f}^2(S) \\
&\geq \sum_{S\subseteq[n],|S|\geq\frac{1}{\gamma}}\big(1-(1-2\gamma)^{|S|}\big)\ \hat{f}^2(S) \\
&\geq \sum_{S\subseteq[n],|S|\geq\frac{1}{\gamma}}\big(1-(1-2\gamma)^{\frac{1}{\gamma}}\big)\ \hat{f}^2(S) \\
&\geq \sum_{S\subseteq[n],|S|\geq\frac{1}{\gamma}}\big(1-e^{-2}\big)\ \hat{f}^2(S)
\end{aligned}
\qquad (3)
$$

Hence,

$$\sum_{S\subseteq[n],|S|\geq\frac{1}{\gamma}}\hat{f}^2(S) \leq \frac{2}{1-e^{-2}}\ ns_\gamma(f) \leq 2.32\ ns_\gamma(f)$$

■

2.2 Application: Learning Half-Space Functions

The above result gives us a couple of interesting corollaries as follows.

Corollary 7 *Consider function $f : \{\pm 1\}^n \to \{\pm 1\}$. Also define function $\beta : [0, 0.5] \to [0, 0.5]$ such that $ns_\gamma(f) \le \beta(\gamma)$ for every $\gamma \in [0, 0.5]$. Then, we have*

$$\sum_{S \subseteq [n], |S| \ge \frac{1}{\beta^{-1}(\frac{\epsilon}{2.32})}} \hat{f}^2(S) \le \epsilon$$

Proof By Theorem 6, we have

$$\sum_{S \subseteq [n], |S| \ge \frac{1}{\beta^{-1}(\frac{\epsilon}{2.32})}} \hat{f}^2(S) \le 2.32 \ \ ns_{\beta^{-1}(\frac{\epsilon}{2.32})}(f) \le 2.32 \ \ \beta(\beta^{-1}(\frac{\epsilon}{2.32})) = \epsilon$$

■

Corollary 8 *Consider half-space function $h : \{\pm 1\}^n \to \{\pm 1\}$. Then, we have*

$$\sum_{S \subseteq [n], |S| \ge \Theta(\frac{1}{\epsilon^2})} \hat{f}^2(S) \le \epsilon$$

Proof From previous lecture, we know that for a half-space function h, we have $ns_\epsilon(h) \le 8.8\sqrt{\epsilon}$. Let $\beta(\epsilon) = 8.8\sqrt{\epsilon}$. Thus, using Corollary 7 we have

$$\sum_{S \subseteq [n], |S| \ge \frac{1}{\beta^{-1}(\frac{\epsilon}{2.32})}} \hat{f}^2(S) \le \epsilon$$

Noting the definition of the $\beta(.)$ function, we have $\beta^{-1}(x) = (\frac{x}{8.8})^2$, and in particular, $\beta^{-1}(\frac{\epsilon}{2.32}) = (\frac{\epsilon}{20.4166})^2 \ge (\frac{\epsilon}{21})^2$. Therefore, $\frac{1}{\beta^{-1}(\frac{\epsilon}{2.32})} \le \frac{441}{\epsilon^2}$. Hence, we can conclude that

$$\sum_{S \subseteq [n], |S| \ge \frac{441}{\epsilon^2}} \hat{f}^2(S) \le \sum_{S \subseteq [n], |S| \ge \frac{1}{\beta^{-1}(\frac{\epsilon}{2.32})}} \hat{f}^2(S) \le \epsilon$$

■

Note that this corollary, along with our low-degree learning algorithm, gives us a $n^{O(\frac{1}{\epsilon^2})}$ sample-uniform learning algorithm for half-spaces. As a side note, we add that one can also learn half-space functions in polynomial time, via linear programming.

2.3 Application: Learning Boolean Functions of k Half-Space Functions

Theorem 9 *Consider half-space functions $h_1, h_2, \ldots, h_k$, and boolean functions $f, g : \{\pm 1\}^k \to \{\pm 1\}$ such that $f(x) = g(h_1(x), h_2(x), \ldots, h_k(x))$. Then, we have $ns_\epsilon(f) \le 8.8k\sqrt{\epsilon}$.*

Proof The main idea of the proof is simple Union bound. We have

$$\begin{aligned}
ns_\epsilon(f) \quad &= \Pr\left[f(x) \ne f(N_\epsilon(x))\right] \\
&= \Pr\left[g(h_1(x), h_2(x), \ldots, h_k(x)) \ne g(h_1(N_\epsilon(x)), h_2(N_\epsilon(x)), \ldots, h_k(N_\epsilon(x)))\right] \\
&\le \Pr\left[h_1(x) \ne h_1(N_\epsilon(x)) \ \ or \ \ h_2(x) \ne h_2(N_\epsilon(x)) \ \ or \ \ \ldots \ \ or \ \ h_k(x) \ne h_k(N_\epsilon(x))\right] \\
&\le \Pr\left[h_1(x) \ne h_1(N_\epsilon(x))\right] + \Pr\left[h_2(x) \ne h_2(N_\epsilon(x))\right] + \ldots + \Pr\left[h_k(x) \ne h_k(N_\epsilon(x))\right] \\
&\le 8.8k\sqrt{\epsilon} \qquad (4)
\end{aligned}$$

where the least inequality follows from Theorem 4. ■

Corollary 10 *Consider half-space functions h_1, h_2, $\ldots$, h_k, and boolean functions $f, g : \{\pm 1\}^k \rightarrow \{\pm 1\}$ such that $f(x) = g(h_1(x), h_2(x), \ldots, h_k(x))$. Then, we have*

$$\sum_{S \subseteq [n], |S| \geq \Theta(\frac{k^2}{\epsilon^2})} \hat{f}^2(S) \leq \epsilon$$

Proof Follows immediately from Corollary 7 and Theorem 9. ■

Again, this corollary, along with our low-degree learning algorithm, gives us a $n^{O(\frac{k^2}{\epsilon^2})}$ sample-uniform learning algorithm for any boolean function of k half-space functions. Side note is that, in contrast to learning half-space functions, it is not known how to learn arbitrary functions of k half-space functions in polynomial time via linear programming.

2.4 Learning Parity Functions

In this model, we assume that there is a black box which contains an unknown parity function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and the goal is to learn this function. Speaking differently, the black box calculates some function $f = \chi_S$ and we want to figure out what S is. For this, we have access to samples of this function, i.e., $(x_1, f(x_1))$, $(x_2, f(x_2))$, $\ldots$, $(x_m, f(x_m))$.

The case without noise: If there is no noise, i.e., we have access to exact evaluations, then this problem can be solved simply by solving a system of linear equations, where we have n unknowns such that the i^{th} unknown is zero if $i \notin S$ and one otherwise.

The case with noise: In the sequel, we focus on the problem of learning parity functions in the presence of noise. This means that for each sample-point x_i, we have $\Pr\left[f(x_i) = \chi_S(x_i)\right] \geq 0.5 + \delta$ where δ is some possibly small positive probability. In this scenario, the goal might be either to find $\chi_S(.)$ that is closest to $f(.)$ or find all $\chi_S(.)$ that are close enough to $f(.)$. The former case is just finding the largest Fourier coefficient while the latter is finding all large enough Fourier coefficients.

There are different variants to this problem. When sample-points x_1, x_2, $\ldots$, x_m are chosen adverserially, the problem is equivalent to *"maximum likelihood decoding of linear codes"* which is known to be NP-hard. When sample-points x_1, x_2, $\ldots$, x_m are chosen randomly at uniform, the problem is equivalent to *"hardness of decoding linear codes"*, or also often called *"hardness of parity with noise"*. A possibly slightly easier version asks what happens if the noise just flips every bit with some probability. This version is often called *"hardness of decoding random linear codes"* which is usually used as an assumption in cryptography and coding. A theorem by Blum, Kalai, and Wasserman gives a slightly sub-exponential ($2^{\frac{n}{\log n}}$) algorithm for this variant. Next time, we will study the question that "what happens if one can make queries into desired sample-points?".