6.895 Randomness and Computation March 3, 2008

Lecture 8

Lecturer: Ronitt Rubinfeld *Scribe: Aleksander Mądry*

Today we visit one of the fundamental problems in Computational Learning Theory: learning parity/linear functions in the presence of noise. In fact, the importance of this problems has been also recognized in many related fields like Coding Theory, Fourier Analysis, and even Cryptography (e.g. there is connection of the hardness of learning noisy parity function to the hardness of certain lattice problems).

1 The model

We assume that there is a black box B_f which contains a circuit computing some parity function $f : \{0,1\}^n \to \{0,1\}$. However, we do not have direct access to B_f. Instead, we can get some (noisy) samples $\langle x_1, \ell_1\rangle, \ldots \langle x_m, \ell_m\rangle$, such that $x_m \in_U \{0,1\}^n$ and ℓ_i is a noisy value of $f(x_i)$, where the nature of the noise i.e. connection between $f(x_i)$ and ℓ_i is to be specified.

2 No noise

Clearly, if $\ell_i = f(x_i)$ for all i, then there is no noise at all—this model correspond to the canonical learning model that was described in previous lectures. In this case, we can easily learn f using $O(n)$ samples in polynomial time. We achieve this, by taking n samples $\langle x_1, f(x_1)\rangle, \ldots, \langle x_n, f(x_n)\rangle$, where all x_i are linearly independent (it can be shown that after taking $O(n)$ samples we will find such linearly independent subset with very high probability) and solving (e.g. by Gaussian elimination) the following linear system:

$$\begin{bmatrix} x_1^1 & \ldots & x_1^n \\ \vdots & \ddots & \vdots \\ x_n^1 & \ldots & x_n^n \end{bmatrix} y = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix},$$

where x_i^j denotes j-th bit of x_i. Note that if $y_* \in \{0,1\}^n$ is a solution to this system then $y_*^i = 1$ iff f contains x_i in its definition i.e. $f(x) = \bigoplus_{i, y_*^i = 1} x_i$. So, we see that learning the parity function when there is no noise is almost trivial.

3 Arbitrary noise

What happens if we let the noise to be arbitrary ? More precisely, what if we assume that we can get only samples $\langle x_1, \tilde{f}(x_1)\rangle, \ldots \langle x_m, \tilde{f}(x_m)\rangle$, where $x_i \in_U \{0,1\}^n$ and $\tilde{f}$ is a function which is only δ-close to f ? Clearly, now the task of learning f can be understood as finding a linear function f' that is closest to $\tilde{f}$. This setting is often referred to in the literature as *agnostic learning* of linear/parity functions, and, since the $\tilde{f}$ can be adversarially chosen, it is thought to be the hardest of models considered in learning.

So, how hard can the problem of agnostic learning of parity functions be? By the machinery that we have already developed, we know that this problem corresponds to finding the largest Fourier coefficient of $\tilde{f}$. And the latter problem is known to be NP-hard. Thus, we are not hoping to obtain a polynomial-time algorithm for this task. We see that this is a significant difference compared to the noiseless case.

4 Random noise

Discouraged by the NP-hardness of the case of arbitrary noise, we can try to relax our problem. Namely, we consider the model in which the noise is random i.e. we have access to samples $\langle x_1, \ell_1 \rangle, \ldots, \langle x_m, \ell_m \rangle$ where independently for each i, $x_i \in_U \{0,1\}^n$ and ℓ_i is equal to $f(x_i)$ with probability $1 - \eta > \frac{1}{2}$. This kind of model is often referred to in the literature as *learning under classification noise*.

Now, it is natural to ask whether parity learning is significantly easier in this model. Somewhat surprisingly, the answer is 'no'. A recent result of Feldman et al. [FGKP06] shows that the problem of agnostic learning can be reduced to the random noise model. Therefore, once again we do not hope for a polynomial-time algorithm.

But still one question remains: can we do better than $2^{O(n)}$ time that is achievable by just brute-force checking all possible linear functions and estimating the distance, by taking enough samples to overcome the effect of noise ?

A positive answer i.e. a slightly sub-exponential, $2^{O(\frac{n}{\log n})}$ algorithm was given by Blum, Kalai, Wasserman [BKW03] and, despite being unimpressive at the first sight, it was used to establish many important results.

5 Blum-Kalai-Wasserman algorithm

Theorem 1 *If $\eta < \frac{1}{2}$, we can learn under classification noise parity function on k inputs in time $2^{O(\frac{k}{\log k})}$.*

Proof

We start the proof of the theorem by establishing the following lemma

Lemma 2 (Noise lemma) *Consider a set of samples $\langle x_1, \ell_1 \rangle, \ldots, \langle x_m, \ell_m \rangle$ corresponding to the noise rate η. Then $Pr[\sum_{i=1}^m \ell_i = f(\sum_{i=1}^m x_i)] = \frac{1}{2} + \frac{1}{2}(1-2\eta)^m$.*

Proof of Lemma 2 We prove the lemma by induction on m. If $m = 1$, then the claim trivially holds. So, let us assume that the claim holds for $m-1$, we will show that it then holds for m. By definition and linearity of f, we have $Pr[\sum_{i=1}^m \ell_i = f(\sum_{i=1}^m x_i)] = Pr[\sum_{i=1}^m \ell_i = \sum_{i=1}^m f(x_i)] = Pr[\ell_m = f(x_m) \wedge \sum_{i=1}^{m-1} \ell_i = \sum_{i=1}^{m-1} f(x_i)] + Pr[\ell_m \neq f(x_m) \wedge \sum_{i=1}^{m-1} \ell_i \neq \sum_{i=1}^{m-1} f(x_i)]$. Now, by noting the fact that noise in each sample is independent and using the inductive hypothesis, we get $Pr[\sum_{i=1}^m \ell_i = f(\sum_{i=1}^m x_i)] = Pr[\ell_m = f(x_m)] \cdot Pr[\sum_{i=1}^{m-1} \ell_i = \sum_{i=1}^{m-1} f(x_i)] + Pr[\ell_m \neq f(x_m)] \cdot Pr[\sum_{i=1}^{m-1} \ell_i \neq \sum_{i=1}^{m-1} f(x_i)] = (1-\eta)(\frac{1}{2} + \frac{1}{2}(1-2\eta)^{m-1}) + \eta(1 - \frac{1}{2} - \frac{1}{2}(1-2\eta)^{m-1}) = \frac{1}{2} + \frac{1}{2}(1-2\eta)^m$.

■

Hence, we see that the sum $\sum_{i=1}^m \ell_i$ has a bias toward the correct value of $f(\sum_{i=1}^m x_i)$. However, this bias vanishes exponentially with the size of the sample set. Therefore, we will need to take a lot of samples in order to amplify this bias to required value.

To show how exactly it will be done, we introduce the following definitions. Let $a = \frac{1}{2}\log k$ and $b = \frac{2k}{\log k}$. Clearly, we have $ab = k$, so these numbers correspond to a division of k-bit string into a blocks of length b each.

Definition 3 *Let $V_i \subseteq \{0,1\}^k$ denote a subspace corresponding to the bitstrings with last i blocks equal to 0 i.e. $x \in V_i$ if and only if $x^j = 0$ for $j > n - i \cdot a$.*

*Moreover, by an i-*sample of size m *we mean a set of m elements of V_i drawn independently with uniform distribution from V_i.*

We now state and prove the following lemma.

Lemma 4 (Progress lemma) *Given i-sample S_i of size m, $x_1, \ldots, x_m$, we can construct in time $O(m \log m)$ an $(i+1)$-sample S_{i+1} of size $\geq m - 2^b$ such that each element of S_{i+1} is a sum of two vectors from S_i.*

Proof of Lemma Let us partition the elements of S_i into (at most 2^b) classes corresponding to possible bit configuration in $(a-i)$-th block of bits. Note that by definition of S_i all blocks after the $(a-i)$-th one are already zeroed out. Now, in each class we pick an arbitrary element, subtract it from all the other elements in the class and throw it away. Next, we take all the remaining elements (after the subtraction) as S_{i+1}. Clearly, this operation can be performed in $O(m \log m)$ time and we have thrown away at most 2^b elements from S_i (one for each class). By construction, each element of S_{i+1} is a sum of two vectors from S_i and, since these two vectors have the same bit configuration in the $(a-i)$-th block, the resulting element of S_{i+1} is from V_{i+1}. Finally, since we have thrown away the elements that we were subtracting, the distribution of elements of S_{i+1} is uniform on V_{i+1} with all elements independent of each other. So, S_{i+1} is an $(i+1)$-sample of size at least $m - 2^b$.

■

It is easy to obtain 0-sample of size m—we simply take m noisy samples and form the set S_0. Now, if we take $m = a \times 2^b$ samples, the Progress Lemma allows us to obtain an $a-1$-sample S_{a-1} of size 2^b. The probability that S_{a-1} contains a basis vector $(1, 0, \ldots, 0)$ is at least $1 - (1 - \frac{1}{2^b})^{2^b} \geq 1 - e^{-1}$, because S_{a-1} has uniform distribution on V_{a-1} and $|V_{a-1}| = 2^b$. So, after repeating the procedure sufficiently many times ($\frac{1}{1-e^{-1}}$ in expectation), we will have $(1, 0, \ldots, 0)$ as a combination of 2^{a-1} elements of S_0. By Noise Lemma, we know that with probability $\frac{1}{2} + \frac{1}{2}(1-2\eta)^{2^{a-1}}$ this combination gives the correct value of $f((1, 0, \ldots, 0))$. So, after repeating the whole procedure $\mathrm{poly}((\frac{1}{1-2\eta})^{2^a})$ times, Hoeffding inequality asserts that the majority vote on the obtained values of the combination gives the correct answer, i.e. the value of $f((1, 0, \ldots, 0))$, with very high probability.

Finally, using the above approach to obtain reliable values of f on the remaining $k-1$ basis vectors, we can use the procedure employed for the noiseless case to obtain f.

Clearly, the running time of the whole algorithm is $\mathrm{poly}((\frac{1}{1-2\eta})^{2^a}, a, 2^b) = \mathrm{poly}((1-2\eta)^{\sqrt{k}}, \log k, 2^{\frac{2k}{\log k}}) = 2^{O(\frac{k}{\log k})}$.

■

6 Noisy parity when we are allowed to make queries

As we have seen in the previous paragraphs, if getting sets of noisy samples of f is all that we can do, obtaining even slightly sub-exponential algorithm was non-trivial. However, if we strengthen our model to allow black-box queries, i.e. ask about a (noisy) value of some particular x_i, then the complexity of the problem changes dramatically.

This is easy to notice in the case of random noise. We can obtain reliable values of f on a basis vector v by just querying the function for sufficiently many pairs x and $x + v$, $\mathrm{poly}(\frac{1}{\eta}, \log n)$ times, and by observing the probability if adding v flips the value of the function.

What is however a bit more surprising is that making queries allows us to learn parity function also in agnostic model. This result, due to Kushilevitz and Mansour [KM91], is another fundamental algorithm with many applications outside Learning Theory. Today we will start describing it.

7 Kushilevitz-Mansour algorithm

The algorithm takes as an input a black box computing f, and parameter $\theta > 0$ (where f is some arbitrary function) and returns a set O of linear functions such that:

- if for some $S \subseteq [n]$, $|\hat{f}(S)| > \theta$ then $\chi_S \in O$ (i.e. we output all linear functions that are close to f)
- if some $\chi_S \in O$ then $|\hat{f}(S)| > \frac{\theta}{2}$ (i.e. we output not too much junk)

As we can see this setting slightly generalizes the agnostic learning in the form presented before.

The algorithm approaches the problem by examining a decision tree being a complete binary tree of height n, whose i-th level corresponds to all 2^i sets $S \subseteq [i]$. Since the size of the tree is exponential, the algorithm must use a very good pruning technique which, after starting from the root, allows it to explore only these subtrees that promise to contain leaves corresponding to the sets $S \subseteq [n]$ with large $|\hat{f}(S)|$.

More precisely, if the algorithm is currently exploring k-th level of the tree and $S_1 \subseteq [k]$ is the currently processed node, then the quantity in which we will be interested is a function $f_{k,S_1} : \{-1,+1\}^{n-k} \to \mathbb{R}$, $f_{k,S_1}(x_{k+1}, \ldots, x_n) = \sum_{T_2 \subseteq \{k_1,\ldots,n\}} \hat{f}(S_1 \cup T_2)\chi_{T_2}(x_{k+1}, \ldots, x_n)$. We should note that in our definition f_{k,S_1} may be not Boolean anymore (even if f is).

Next time we will devise a method for using f_{k,S_1} as a guide in our exploration of the decision tree.

References

[BKW03] A. Blum, A. Kalai, and H. Wasserman. *Noise-tolerant learning, the parity problem, and the statistical query model.* Journal of the ACM, :50(4):506-519, 2003.

[FGKP06] V. Feldman, P. Gopalan, S. Khot, and A. Ponnuswami *On Agnostic Learning of Parities, Monomials and Halfspaces.* In Proceedings of FOCS '06.

[KM91] E. Kushilevitz, and Y. Mansour. *Learning decision trees using the Fourier spectrum.* In Proceedings of STOC '91