6.842 Randomness and Computation March 17, 2014

Lecture 12

Lecturer: Ronitt Rubinfeld *Scribe: Alex Wein*

1 Lecture Overview

This lecture will cover learning. Topics will include:

- Learning from random examples
- PAC (probably approximately correct) learning
- Brute force learning algorithm
- Efficient learning algorithm: conjunctions
- Begin: learning via Fourier representation

2 Learning from random examples

Suppose there is an unknown function $f : D \to R$ that we wish to learn. We are given access to an example oracle $\text{Ex}(f)$ which generates m i.i.d. samples $x_1, \ldots, x_m$ from a given distribution $\mathcal{D}$ on D, and outputs $(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_m, f(x_m))$. We (the learner) must output a hypothesis function $h : D \to R$ that should be similar to f. Note that the learner is not allowed to choose x values and query the corresponding $f(x)$. Instead, the learner must work with whatever random examples the example oracle happens to generate.

The following two related definitions formalize the notion of h being "close to" f.

Definition 1 *The* error *of h with respect to f is* $\text{error}(h) = \Pr_{x \in \mathcal{D}}[h(x) \neq f(x)]$.

Definition 2 *h is* ε-close *to f with respect to $\mathcal{D}$ if* $\Pr_{x \in \mathcal{D}}[f(x) \neq h(x)] \leq \varepsilon$.

The term *error* is generally used in learning whereas the term *ε-close* is generally used in property testing, but both refer to the same concept. Note the importance of the distribution $\mathcal{D}$. We can only expect our learning algorithms to do well when the distribution used to measure the error is the same as the distribution used to generate the examples. Today we will take $\mathcal{D}$ to be the uniform distribution.

Now we formally define the learning problem based on the PAC (probably approximately correct) framework. We cannot hope to learn if f can be an arbitrary function, so we assume f belongs to a known class of functions $\mathcal{C}$ called the *concept class*.

Definition 3 *A* uniform distribution learning algorithm *for a concept class $\mathcal{C}$ is an algorithm $\mathcal{A}$ such that*

- *$\mathcal{A}$ is given $\varepsilon, \delta > 0$ and access to* $\text{Ex}(f)$ *for some $f \in \mathcal{C}$. (Here, $\mathcal{C}$ is known but f is unknown.)*
- *$\mathcal{A}$ outputs h such that with probability $\geq 1-\delta$,* $\text{error}(h)$ *with respect to f is at most ε (or equivalently, h is ε-close to f).*

The following parameters are of interest when designing a learning algorithm.

- Runtime

- Sample complexity (number of examples used)
- Accuracy paramter ε
- Confidence parameter δ
- Description of h
- Efficiency of evaluating h

Note that it is not obvious how the output function h should be described. When $h \in \mathcal{C}$ and h is described in the same way that f is then we have *proper learning*. We generally want the description of h to have length $\mathcal{O}(\log |\mathcal{C}|)$ and clearly it is impossible to do better.

It is possible (but somewhat involved) to perform error reduction in order to arbitrarily decrease the confidence parameter δ. The runtime scales as $\mathcal{O}(\log \frac{1}{\delta})$.

3 Brute force learning algorithm

In this section we will see that the issue is not the sample complexity but the runtime. In particular we will give a simple learning algorithm that uses only a few samples but has bad runtime. This construction works for any distribution $\mathcal{D}$ but we will take $\mathcal{D}$ to be uniform here. The algorithm is as follows.

- Draw $M = \frac{1}{\varepsilon}(\ln |C| + \ln \frac{1}{\delta})$ uniform samples.
- Search over all $h \in \mathcal{C}$ until you find one that is consistent with all M examples, and output it. (Choose arbitrarily if there is more than one such h.)

To analyze the behavior of this algorithm, note that f is consistent with all examples and so something gets output. We need to show that whatever h gets output has small error. Call h "bad" if it is ε-far from f (i.e. not ε-close). The probability that a particular bad h is consistent with all M examples is at most $(1-\varepsilon)^M$ because h and f differ on at most ε fraction of inputs. By the union bound, the probability that *any* bad $h \in \mathcal{C}$ is consistent with all examples is at most

$$|\mathcal{C}|(1-\varepsilon)^M = |\mathcal{C}|(1-\varepsilon)^{\frac{1}{\varepsilon}(\ln |\mathcal{C}| + \ln \frac{1}{\delta})} \leq |\mathcal{C}|(e^{-1})^{\ln |\mathcal{C}| + \ln \frac{1}{\delta}} \leq \delta.$$

Here we have used the identity $\ln(1+x) \leq x$ in order to conclude $\ln(1-\varepsilon) \leq -\varepsilon \Rightarrow \frac{1}{\varepsilon}\ln(1-\varepsilon) \leq -1 \Rightarrow (1-\varepsilon)^{1/\varepsilon} \leq e^{-1}$. Therefore the algorithm outputs a good h with probability $\geq 1-\delta$.

One problem to watch out for is if $\mathcal{C}$ is too big then you will need to take many samples before you can rule out all bad h's. One example of this is the "Bible code", a purported set of secret messages hidden in the Bible. If you try enough different ways of extracting a secret message you are bound to find one that works by chance. Similarly, if you look through enough statistical studies you are bound to find some that are wrong, allowing you to claim a false result.

4 Efficient learning algorithm: conjunction

Let $\mathcal{C}$ be the conjunctions over $\{0,1\}^n$, e.g. $f(x) = x_i x_j \bar{x}_k$. A conjunction is a single "and" of variables and/or their complements. Note that there are 3^n different possible conjunctions because for each i, the conjunction can contain x_i, $\bar{x}_i$, or neither. We might first wonder whether it is possible to achieve perfect accuracy ($\varepsilon = 0$) using subexponentially-many samples. It turns out this is impossible because there is

no way to distinguish the zero function $f(x) = 0$ from a conjunction of the form $f(x) = \wedge_i x_i^{b_i}$ where b_i indicates whether x_i is complemented or not. The reason you need exponentially-many samples to distinguish these two functions is because the second one only evaluates to 1 on one of the 2^n possible inputs.

The following is an efficient learning algorithm for conjunctions.

- Draw poly$(\frac{1}{\varepsilon})$ random examples and estimate $\Pr[f(x) = 1]$ to additive error $\pm\frac{\varepsilon}{4}$. If the estimate is $< \frac{\varepsilon}{2}$ then output $h(x) \equiv 0$.
- Since the estimate is $\geq \frac{\varepsilon}{2}$ with error $\frac{\varepsilon}{4}$, we must have $\Pr[f(x) = 1] \geq \frac{\varepsilon}{4}$. This means we expect a new random positive example (i.e. $f(x) = 1$) every $\mathcal{O}(\frac{1}{\varepsilon})$ examples.
- Consider only the set of positive examples. Let V be the set of indices i such that x_i is set the same way in every positive example. Output $h(x) = \wedge_{i \in V} x_i^{b_i}$ where b_i indicates whether or not x_i was complemented in every positive example.

To analyze the behavior of this algorithm, note that if $x_i^{b_i}$ appears in f then it will be set the same way in every positive example and will therefore appear in h. If $x_i^{b_i}$ does not appear in f then $\Pr[i \in V] \leq \frac{1}{2^{k-1}}$ where k is the number of positive examples, because each of the positive examples sets x_i to 0 or 1 uniformly and independently. By the union bound, the probability that any i not in f survives is $\leq \frac{n}{2^{k-1}}$. This means it is sufficient to run the algorithm until we have seen $k = 1 + \log_2 \frac{n}{\delta}$ positive examples. The probability of failure is $\leq \frac{n}{2^{k-1}} = \delta$. Note that the total number of samples used is $\mathcal{O}(\frac{1}{\varepsilon} \log \frac{n}{\delta})$. This is even better than the brute force algorithm, which would require $M = \frac{1}{\varepsilon}(\ln 3^n + \ln \frac{1}{\delta}) = \mathcal{O}(\frac{1}{\varepsilon}(n + \log \frac{1}{\delta}))$.

5 Learning via Fourier representation

We will next be talking about learning functions that are sparse in the Fourier domain, i.e. most Fourier coefficients are small. Such functions can be well-approximated by only the largest few coefficients. Our first result will be that it is easy to approximate a single Fourier coefficient of the unknown function f.

Lemma 4 *It is possible to approximate any one Fourier coefficient S to within additive error γ (i.e. $|output - \hat{f}(S)| \leq \gamma$) with probability $\geq 1 - \delta$ using $\mathcal{O}(\frac{1}{\gamma^2} \log \frac{1}{\delta})$ samples.*

The idea of the proof is to use the fact from last time that $\hat{f}(S) = 2 \Pr_x[f(x) = \chi_S(x)] - 1$ and to estimate the probability $\Pr_x[f(x) = \chi_S(x)]$ using the Chernoff bound or Hoeffding's inequality. Note that this requires only a random example oracle rather than the ability to query specific values of x.

Although it is easy to approximate a single Fourier coefficient of f, it is much harder to find the largest Fourier coefficients of f. More on this next time.