CS 590 Sublinear algorithms Jan, 2015

Lecture 9

Lecturer: Abhiram Natarajan *Scribe(s): Abhiram Natarajan*

0 Outline

This lecture begins with a very brief introduction to computational learning theory. We then look at some connections between property testing and learning theory. Finally, we shall look at the problem of density estimation for discrete distributions.

1 Introduction

Wikipedia defines computational learning theory (CLT) as the intersection of machine learning and theoretical computer science [13]. Computational understanding of intelligence has been the subject of intense research for more than four decades. In a way, it is the next frontier for science. CLT approaches the problem with the methodology of theoretical computer science, i.e., abstracting out problem incidentals, making minimal assumptions and proceeding with rigour and with adherence to accepted mathematical standards of proof. It is hoped the vantage point afforded by CLT will allow us to get very deep insights into the nature of intelligence and learning. The fact that fundamental advances in learning theory have, both directly and indirectly, influenced the tremendous success of intelligent systems is what provides grist to this seemingly bombastic claim.

2 Rectangle Learning Game

In this section, we consider a simple learning problem. We look at an algorithmic solution to the problem and get some exposure to the framework under which learning algorithms are analyzed. All the material below is from Kearns and Vazirani's excellent introduction to CLT [8].

2.1 Description

Consider a two-player game where the players are nature and a learner. The nature knows of an axis aligned rectangle and is willing to reveal tidbits of information about the rectangle. The goal of the learner is to learn the rectangle as accurately as possible.

More formally, the setting of the problem is as follows:

- (hidden, known to nature): An axis aligned rectangle R in $\mathbb{R}^2$. We can think of R as a function, i.e. $R : \mathbb{R}^2 \to \{0, 1\}$, where $R(x) = 1$ iff x is inside the rectangle or is on the boundary of the rectangle. Also, there is a distribution D on the points in $\mathbb{R}^2$.
- (available to learner): Access to labelled examples $< x, R(x) >$, where x is chosen by nature according to D.

- (what we want): An algorithm that outputs a hypothesis R'[1] such that R' is close to R according to the distribution D. In other words, for $x \sim D$, we want the probability of $R'(x) \neq R(x)$ to be low.

2.2 Algorithm

Algorithm 1 is a simple algorithm that solves the problem.

Algorithm 1 : Output a hypothesis rectangle

```
1: take $m$ labeled samples
2: output the smallest tightest fit axis aligned rectangle that contains all the $x$ values for which
   $R(x)$ was 1
```

2.3 Analysis

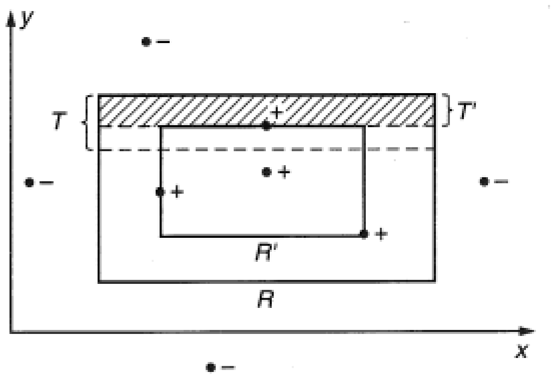

Figure 1: The target rectangle R, the hypothesis R' and labelled examples

On the surface, Algorithm 1 appears to be a reasonably good attempt. However, it does not give any indication whatsoever as to what value of m would allow us to get an accurate rectangle. To this end, we now introduce a widely embraced notion for studying the efficacy of a learning algorithm, namely Probably-Approximately-Correct (PAC) learning. PAC is a mathematical framework introduced by Valiant [11]. To quote Haussler [4]

> ...the intent of the PAC model is that successful learning of an unknown target concept should entail obtaining, with high probability, a hypothesis that is a good approximation of it...

In other words, we want our algorithm to have *small error with high confidence.* Mathematically, given small values $0 \leq \epsilon, \delta \leq \frac{1}{2}$, we want that

$$P(error > \epsilon) \leq \delta$$

[1]Remember we will never know the exact rectangle unless we take ∞ samples, so any learning algorithm which terminates in finite time only has a *hypothesis* of the true rectangle.

In this problem, we define error as follows

$$error := P_{x \sim D}(R(x) \neq R'(x))$$

Figure 1 helps us visualize what is going on.[2] As we can see, we have chosen R' to be such that it is the smallest possible axis aligned rectangle that encompasses all positively labelled examples. To analyze the probability of error of this algorithm, let us sweep out an area T such that its weight under D is $\frac{\epsilon}{4}$. Remember, our intention is to get a sample from this area (because getting a sample from this area would mean that R' would include this part). The un-favourable event here would be if we were to never obtain a sample from T. What is the probability of a particular point not being from T? It is just $1 - \frac{\epsilon}{4}$. To not have R' encroach on T, we would need all our m samples to not be from T. The probability of this event is $\left(1 - \frac{\epsilon}{4}\right)^m$. Thus, we have deduced that:

$$P(\text{having all our } m \text{ sample points miss } T) = \left(1 - \frac{\epsilon}{4}\right)^m$$

By symmetry, if we envision regions similar to T on all sides of R, we would get the probability of missing each T-like region to be the same. Thus we have:

$$\begin{aligned} P(\text{having an error } \epsilon) &\leq P(\text{having an error of } \frac{\epsilon}{4} \text{ on any side of } R) && \text{[erring on the side of caution]} \\ &\leq 4 * P(\text{having an error of } \frac{\epsilon}{4} \text{ on one side of } R) && \text{[By union bound]} \\ &= 4\left(1 - \frac{\epsilon}{4}\right)^m \end{aligned}$$

We wish to have $4\left(1 - \frac{\epsilon}{4}\right)^m \leq \delta$. Using the commonly used inequality $1 - x \leq e^{-x}$, we find that having $m = \frac{4}{\epsilon} \log \frac{4}{\delta}$ gives us the results we desire.

Let us take a closer look at the sample complexity. It is polylog of $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$. This is good news. We can now say that we can learn axis aligned rectangles in an efficient PAC manner. What we just proved was the following theorem.

Theorem 1 *There exists an efficient PAC learning algorithm to learn axis aligned rectangles.*

3 Connections between Learning Theory and Property Testing

It should already be clear that there is some equivalence between the central goals of learning and the central goals of property testing. In this section, we shall look at this connection in more detail. The material below is from the work of Goldreich et al. [3].

3.1 Re-defining Property Testing

Here, we will introduce a few more parameters into the canonical property testing problem that will provide some flexibility. This will help us draw analogies with learning problems more

[2]Figure from `http://www.cs.ucsb.edu/~suri/cs231/suriLearning.pdf`

clearly.

Let there be some property P, and let $\mathcal{F}$ be the class of functions that satisfy P. Also, assume the presence of a metric d that is used to quantify the *distance* between two functions.[3] Now, testing if some function f has the property P is equivalent to testing whether $f \overset{?}{\in} \mathcal{F}$. More formally, given small ϵ, δ, and a distribution D, we want to distinguish between the following cases:

- (YES case) $f \in \mathcal{F}$
- (NO case) $d(f, \mathcal{F}) \geq \epsilon$ - note that d is measured w.r.t. D - where

$$d(f, \mathcal{F}) := \inf_{g \in \mathcal{F}} d(f, g)$$

We need that

$$P(\text{algorithm returns the right output}) \geq 1 - \delta$$

With this definition, let us compare and contrast property testing and learning theory.

	Property Testing	Learning Theory
1	$\mathcal{F}$ is known	$\mathcal{F}$ is known
2	has to handle any possible f	has to handle only $f \in \mathcal{F}$
3	has to output just "YES" or "NO"	has to output an actual function $\hat{f}$ which is hopefully in $\mathcal{F}$

Table 1: Similarities and Differences b/w property testing and learning

Table 1 contains some high level comparisons between learning and property testing. Looking more carefully at point 2, it seems that property testing has to be able to handle a much wider range of functions. It is completely possible that all the functions in $\mathcal{F}$ are easy to work with, so as such, learning algorithms might have an easier time dealing with certain kinds of input. So one gets the impression that learning should be easier than property testing. On the other hand, as point 3 suggests, property testing algorithms have to output just a "YES" or a "NO". In comparison, a learning algorithm has to actually output a function, which is hopefully in $\mathcal{F}$. This makes the task of learning appear harder than property testing.

It turns out that there is no answer in either direction that holds for all cases. We shall now see some simple arguments that suggest that neither kind of problem is always harder than the other. This is in contrast to similar questions about the relative complexity of the decision version of a problem (e.g. Can this MAX-SAT have an answer with more than k clauses satisfied?), its search version (e.g. What is the maximum value of k such that at least k clauses can be satisfied in this particular MAX-SAT instance?), and its exact computation version (e.g. What is the maximal satisfying assignment for this instance of MAX-SAT?). Here there is a clear answer. Please read Madhu Sudan's lecture notes [9] if you want more details about this.

[3]For instance, if we are considering two boolean-valued boolean functions f, g, a possible distance notion could be the fraction of inputs on which they disagree. This would be denoted as $d(f, g) := P_{x \sim D}[f(x) \neq g(x)]$ where D could be any distribution over the input D.

3.2 Direct Comparisons b/w Learning and Testing

Before we begin, we must define what proper learning is. A learning algorithm is a proper learning algorithm if it can promise that every hypothesis function that it outputs will belong to the same function class as the target function f, i.e. the hypothesis function $\hat{h} \in \mathcal{F}$.

We also state the additive forms of the Chernoff bound which we shall need. Let $X_1, \dots, X_m$ be m independent random variables where $X_i \in [0,1]$. Let $p := \frac{1}{m}\sum_{i=1}^{m} \exp[X_i]$. For every $\gamma \in [0,1]$,

$$P\left(\frac{1}{m}\sum_{i=1}^{m} X_i > p + \gamma\right) \leq e^{-2\gamma^2 m}$$

and

$$P\left(\frac{1}{m}\sum_{i=1}^{m} X_i < p - \gamma\right) \leq e^{-2\gamma^2 m}$$

3.2.1 Testing is not harder than proper learning

Proposition 2 *Let a function class $\mathcal{F}$ have a proper learning algorithm $\mathcal{A}$. Then $\mathcal{F}$ has a property testing algorithm $\mathcal{A}'$ with sample complexity*

$$\mathfrak{m}_{\mathcal{A}'}(n, \epsilon, \delta) = \mathfrak{m}_{\mathcal{A}}\left(n, \frac{\epsilon}{2}, \frac{\delta}{2}\right) + O\left(\frac{\log\left(\frac{1}{\delta}\right)}{\epsilon}\right)$$

Proof

Algorithm 2 : $\mathcal{A}'$ - Test for membership in $\mathcal{F}$ using $\mathcal{A}$

```
1: Run $\mathcal{A}$ with $\frac{\delta}{2}$ and $\frac{\epsilon}{2}$ as the parameters
2: if $\mathcal{A}$ returns "NO FUNCTION" then
3:   return "ACCEPT"
4: else
5:   $h \leftarrow$ output of $\mathcal{A}$
6:   draw an additional sample of size $O\left(\frac{1}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$
7:   approximate distance between $h$ and $\mathcal{F}$ using the additional sample as $dist$
8:   if $dist < \frac{\epsilon}{4}$ then
9:     return "ACCEPT"
10:  else
11:    return "REJECT"
12:  end if
13: end if
```

Let us analyze Algorithm 2. If $f \in \mathcal{F}$, then with prob at least $1 - \frac{\delta}{2}$, $\mathcal{A}$'s output h is $\frac{\epsilon}{2}$-close to f. Using the additive Chernoff bound that we just saw, we can see that with probability at least $1 - \frac{\delta}{2}$ over the additional sample, we shall not reject it. If f is ϵ-far from $\mathcal{F}$, any hypothesis

$h \in \mathcal{F}$ is at least ϵ-far from f, and with probability at least $1 - \frac{\delta}{2}$ over the additional sample, f is rejected. ■

3.2.2 Testing may be harder than learning

Section 1.5 of Kearns and Varizani [8] shows us that by using carefully chosen *representation classes*, we can sometimes get efficient learning algorithms for otherwise intractable problems. For example, they show that any 3-term DNF formula can be written as a 3-CNF formula, and that we can PAC learn 3-term DNF formulae if we allow they hypothesis to be represented as a 3-CNF formula, but not if we insist that it be represented as a 3-term DNF formula. To prove that testing may be harder than learning, we shall show for these problems, we can never have an efficient property testing algorithm. The strategy of the proof will be to show that by using an efficient property testing algorithm, we can create an decision procedure that contradicts the intractability of the problem under consideration.

More specifically, let the following decision problem be NP-complete. You are given $(x_1, \sigma_1), \ldots (x_t, \sigma_t)$ as input where σ_i is the result of applying some function on x_i. The question is if $\overset{?}{\exists} f \in \mathcal{F}$ s.t. $f(x_i) = \sigma_i \; \forall i$? Just to re-iterate, this decision problem is NP-complete. Basically, we want to find a function f that would be consistent with the given input. We shall now show that with a clever choice of the error parameter, we can use a property testing algorithm for $\mathcal{F}$ to get an answer to this question very efficiently. Assuming the truth of the conjecture $NP \not\subset BPP$, this deduction forms a violation. Thus it would mean that an efficient property testing algorithm for this sort of question cannot exist.

Let us assume the presence of an efficient tester for the class $\mathcal{F}$. The tester takes as input ϵ, δ and m labeled samples and returns "YES" or "NO" based on whether there is an $f \in \mathcal{F}$ that is consistent with the input samples. Since we are talking about *efficient* testers, the tester will take time $poly\left(\frac{n}{\epsilon}\right)$ to return its answer as to whether there exists an $f \in \mathcal{F}$ such that $P(d(f, \mathcal{F}) \geq \epsilon) \leq \delta$. How can we use such a tester to solve the NP-complete decision problem?

The trick is very simple. Set the error parameter ϵ of the property testing algorithm to be $\frac{1}{2t}$. Also, let $D \sim Unif([t])$, i.e. D is the uniform distribution on the set $[t] = \{1, 2, \ldots t\}$. Additionally, choose δ to be very low. Now, if we indeed had an efficient property testing algorithm, it would be able to determine *with perfect certainty* as to whether there exists an $f \in \mathcal{F}$ that is perfectly consistent with the given input. Why is that? It is because we have chosen our error parameter to be $\frac{1}{2t}$. This prevents the algorithm from allowing any error whatsoever while ascertaining the existence of a suitable f in $\mathcal{F}$, because we have chosen an uniform distribution over $[t]$ (even a single error would mean that the probability of error $\geq \frac{1}{t}$, which is not allowed because of our carefully chosen error parameter). Because solving the intractable problem is completely equivalent to verifying $\overset{?}{\exists} f \in \mathcal{F}$ that is consistent with $(x_1, \sigma_1) \ldots (x_t, \sigma_t)$, if we indeed had an efficient property testing algorithm, we could have solved the NP-complete decision problem efficiently. This contradicts our belief that $NP \not\subset BPP$. This completes the proof of the claim.

3.2.3 Testing may be easier than learning

Please read Section 3.3. of Goldreich et al. [3].

4 Discrete Density Estimation

Density estimation is the problem of constructing an estimate of an unknown underlying probability distribution from which samples are available. Let us consider the problem of density estimation over a discrete domain $[d]$.[4] The material below is from the following references [2,6,7].

4.1 Algorithm

We are given an unknown distribution $\mathbf{p} = (p_1, p_2, \ldots, p_d)$ over $[d]$, with the obvious constraint that $\sum_{i=1}^{n} p_i = 1$, Algorithm 3 outputs a hypothesis distribution $\hat{\mathbf{p}}$. $\hat{\mathbf{p}}$, as described in the algorithm, is normally just called the empirical distribution. Essentially, the probability of an element $i \in [d]$ is just taken to be the fraction of occurrences of the element in the m values obtained.

Algorithm 3 : Output a hypothesis distribution

```
1: take $m$ samples, $x_1 \ldots x_m$, from $\mathbf{p}$
2: return $\hat{\mathbf{p}} = (\hat{p}_1, \ldots, \hat{p}_d)$ such that
       $$\hat{p_i} = \frac{X_i}{m}$$
   where $X_i = \sum_{j=1}^{m} \mathbf{1}_{x_j = i}$
```

4.2 Algorithm Analysis

We study the ℓ_1 distance between the hypothesis distribution and the underlying distribution. This is because the ℓ_1 distance is very robust and other distribution distance notions often have connections to the ℓ_1 distance. The ℓ_1 distance between $\hat{\mathbf{p}}$ and $\mathbf{p}$ is denoted as $\|\hat{\mathbf{p}} - \mathbf{p}\|_1$ and it is just equal to $\sum_{i=1}^{d} |\hat{p}_i - p_i|$. Since $\hat{\mathbf{p}}$ is a random variable (random vector to be precise), we shall actually just study the expected value of the ℓ_1 distance. Note that upper bounding the expected value of the error allows us to get a *PAC-like* assertion ($P(error \geq \epsilon) \leq$ something) by simple use of Markov's inequality.

4.2.1 Upper Bound

Consider the random variable X_i. It is the summation of m Bernoulli random variables each of which is 1 whenever the sample is i and 0 otherwise. Thus, we can say that each X_i is a binomial random variable, i.e. $X_i \sim Bin(m, p_i)$.

[4] $[d] = \{1, \ldots, d\}$

$$
\begin{aligned}
\mathbb{E}[|X_i - mp_i|] &= \sqrt{\mathbb{E}[|X_i - mp_i|]^2} \\
&= \sqrt{\mathbb{E}[(X_i - mp_i)^2] - Var[|X_i - mp_i|]} \\
&\leq \sqrt{\mathbb{E}[(X_i - mp_i)^2]} \\
&= \sqrt{Var[X_i]} \\
&= \sqrt{mp_i(1-p_i)} \\
\implies \frac{\mathbb{E}[|X_i - mp_i|]}{m} &\leq \sqrt{\frac{p_i(1-p_i)}{m}} \qquad (1)
\end{aligned}
$$

Equation 1 gives us $\mathbb{E}[|\hat{p_i} - p_i|] \leq \sqrt{\frac{p_i(1-p_i)}{m}}$. By linearity of expectation, we have that

$$
\mathbb{E}[\|\hat{\mathbf{p}} - \mathbf{p}\|_1] = \sum_{i=1}^{d} \mathbb{E}[|\hat{p_i} - p_i|] \leq \sum_{i=1}^{d} \sqrt{\frac{p_i(1-p_i)}{m}} \leq \frac{1}{\sqrt{m}} \sqrt{\sum_{i=1}^{d} (\sqrt{p_i})^2} \times \sqrt{\sum_{i=1}^{d} \left(\sqrt{(1-p_i)}\right)^2} \leq \sqrt{\frac{d}{m}}
$$

where the second inequality follows due to the Cauchy-Schwartz inequality. This shows that with $m \in \Theta\left(\frac{d}{\epsilon^2}\right)$, we can get $\mathbb{E}[\|\hat{\mathbf{p}} - \mathbf{p}\|_1] \leq \epsilon$.

4.2.2 Lower Bound

The right way to prove the lower bound would be to define two distributions, $\mathbf{P}_1$ which has $\frac{d}{2}$ elements with probability $\frac{1-\epsilon}{d}$, and another half each with probability $\frac{1+\epsilon}{d}$, and $\mathbf{P}_2$, which has the first half elements with a mass of $\frac{1+\epsilon}{d}$, and the second half each with mass of $\frac{1-\epsilon}{d}$. The ℓ_1 distance between these two distributions is 2ϵ. Based on the well known reduction from estimation to testing [10], we then have to show that we need $m \in \Omega\left(\frac{d}{\epsilon^2}\right)$ to be able to test between these distributions with error $\leq \epsilon$. In order to show that you need a minimum number of samples be able to test between these two distributions, you need a minimax lower bound argument. For details, please refer to these lecture notes scribed by Jiantao Jiao [5].

Here, I shall just give a heuristic explanation about why this could be true. Consider a distribution over $[d]$ where every pair of points $2i$ and $2i+1$ have probability masses of $\frac{1-\epsilon}{d}$ and $\frac{1+\epsilon}{d}$. The order is not known, i.e. it could be $p_{2i} > p_{2i+1}$ or vice-versa. Thus, intuitively, the problem of density estimation cannot be simpler than working with $\frac{d}{2}$ slightly biased coins, and with us having to figure out the more biased side of each coin.

From Wikipedia's page on the Chernoff bound [12], we have the following proposition.

Proposition 3 *Given a coin with bias $p > \frac{1}{2}$, assuming the side which turns up with probability p is unknown, consider the following procedure - you flip it many times and then choose the side that comes up the most. The number of times you have to flip it to have the error probability to be at most ϵ, is at least*

$$
\frac{1}{\left(p - \frac{1}{2}\right)^2} \ln \frac{1}{\sqrt{\epsilon}}
$$

Using this, we see that we need at least $\Omega\left(\frac{1}{\epsilon^2}\right)$ samples to handle every pair of elements. Thus, on the whole, we would need $\Omega\left(\frac{d}{\epsilon^2}\right)$ samples.

Finally, we have proved the following theorem.

Theorem 4 *Sample size of $\Omega\left(\frac{d}{\epsilon^2}\right)$ is necessary and $\Theta\left(\frac{d}{\epsilon^2}\right)$ is sufficient to learn discrete probability distributions over $[d]$ to an error of at most ϵ.*

4.3 Why did we talk about density estimation in a sub-linear algorithms course?

We just saw that for any arbitrary distribution over a finite sized discrete domain, the sample complexity scales linearly in the number of dimensions. This suggests that any algorithm that works for this general case essentially handles each dimension separately.

An obvious question now is whether we can do better when there is some sort of structure in our distribution. It turns out that for a lot of naturally occurring discrete distributions, you can have the sample complexity to be sub-linear in the number of dimensions. Examples of such distributions are monotone distributions, k-modal distributions and log-concave distributions. Very recently, it was found that sample complexity to learn Poisson Binomial Distributions up to arbitrary accuracy is constant in the size of the domain! The next lecture is going to be exclusively about this rather surprising result.

Please refer to this talk by Daskalakis [1] for more information and related references.

References

[1] Constantinos Daskalakis. Beyond berry-esseen: Structure and learning sums of random variables, 2014.

[2] Ilias Diakonikolas. Beyond histograms: Exploiting structure in distribution estimation, 2014.

[3] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM (JACM)*, 45(4):653–750, 1998.

[4] David Haussler. *Probably approximately correct learning.* University of California, Santa Cruz, Computer Research Laboratory, 1990.

[5] Jiantao Jiao. Minimax lower bounds and minimax-optimal estimation, 2013.

[6] Jiantao Jiao. Personal Communication, 2015.

[7] Jiantao Jiao, Kartik Venkat, and Tsachy Weissman. Order-optimal estimation of functionals of discrete distributions. *CoRR*, abs/1406.6956, 2014.

[8] Michael J Kearns and Umesh Virkumar Vazirani. *An introduction to computational learning theory.* MIT press, 1994.

[9] Madhu Sudan. 6.893: Approximability of optimization problems (lecture 8), 1999.

[10] Alexandre B. Tsybakov. *Introduction to Nonparametric Estimation.* Springer Publishing Company, Incorporated, 1st edition, 2008.

[11] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[12] Wikipedia. Chernoff bound — wikipedia, the free encyclopedia, 2015. [Online; accessed 15-February-2015].

[13] Wikipedia. Computational learning theory — wikipedia, the free encyclopedia, 2015. [Online; accessed 14-February-2015].