6.080/6.089 GITCS Feb 5, 2008

Lecture 21

Lecturer: Scott Aaronson *Scribe: Scott Aaronson / Chris Granade*

1 Recap and Discussion of Previous Lecture

Theorem 1 (Valiant) $m = O\left(\frac{1}{\epsilon}\log\left(|C|/\delta\right)\right)$ *samples suffice for* (ϵ, δ)*-learning.*

Theorem 2 (Blumer et al.) $m = O\left(\frac{1}{\epsilon}\,\mathrm{VCdim}\,(C)\log\frac{1}{\delta\epsilon}\right)$ *samples suffice.*

In both cases, the learning algorithm that achives the bound is just "find any hypothesis h compatible with all the sample data, and output it."

You asked great, probing questions last time, about what these theorems really mean. For example, "why can't I just draw little circles around the 'yes' points, and expect that I can therefore predict the future?" It's unfortunately a bit hidden in the formalism, but what these theorems are "really" saying is that to predict the future, it suffices to find a succinct description of the past–a description that takes many fewer bits to write down than the past data itself. Hence the dependence on $|C|$ or $\mathrm{VCdim}\,(C)$: the size or dimension of the concept class from which our hypothesis is drawn.

We also talked about the computational problem of finding a small hypothesis that agrees with the data. Certainly we can always solve this problem in polynomial time if $\mathsf{P} = \mathsf{NP}$. But what if $\mathsf{P} \neq \mathsf{NP}$? Can we show that "learning is NP-hard"? Here we saw that we need to distinguish two cases:

Proper learning problems (where the hypothesis has to have a certain form): Sometimes we can show these are NP-hard. Example: Finding a DNF expression that agrees with the data.

Improper learning problems (where the hypothesis can be any Boolean circuit): It's an open problem whether any of these are NP-hard. (Incidentally, why do we restrict the hypothesis to be a Boolean circuit? It's equivalent to saying, we should be able to compute in polynomial time what a given hypothesis predicts.)

So, if we can't show that improper (or "representation-independent") learning is NP-complete, what other evidence might there be for its hardness? The teaser from last time: we could try to show that finding a hypothesis that explains past data is at least as hard as breaking some cryptographic code!

But how would we actually do this? How would we reduce a cryptanalysis problem to a learning problem? To be concrete, let's just consider the RSA cryptosystem. Can any of you give me a PAC-learning problem, such that if you could solve it, then you could also break RSA?

How about this: our concept class C will have one concept c for each product of prime numbers $N = pq$, with $p - 1$ and $q - 1$ not divisible by 3. (Technically, for each N expressible with at most n bits.)

Our sample space S will consist of pairs of the form (y, i), where $1 \leq y \leq N-1$ and $1 \leq i \leq \log N$. Here's the trick: (y, i) will be in c if and only if the i^{th} bit of $y^{1/3} \bmod N$ is a 1. The sample distribution D will be uniform over S.

So basically, you (the learner) will be given a bunch of encrypted messages of the form $x^3 \bmod N$, and for each one, you'll also told some bit of the plaintext x. Based on this "training" data, you need to infer the general rule for going from $x^3 \bmod N$ to some given bit of x.

First question: is there such a rule, which is expressible by a polynomial-size circuit? Sure there is! Namely, the rule that someone who knew the trapdoor information, who knew p and q, would use to decrypt the messages!

On the other hand, if you don't already know this rule, is there an efficient algorithm to infer it from sample data? Well, not if RSA is secure! The sample data–the set of (y, i) pairs–is stuff that an eavesdropper could not only plausibly have access to, but could actually generate itself! So if, by examining that data, the adversary could gain the ability to go from $x^3 \bmod N$ to a desired bit of x—well, then RSA is toast. (Today, it's actually known how to base the hardness of improper learning on the existence of any one-way function, not just the RSA function.) A beautiful connection between learning theory and cryptography—typical of the connections that abound in theoretical computer science.

1.1 RSA and Language Learning In Infants: The Argument Chomsky Should've Made

What is Noam Chomsky famous for, besides hating America? Linguistics, of course—among other things, what's called the "Poverty of the Stimulus Argument." This is an argument that tries to show, more or less from first principles, that many of the basic ingredients of grammar (nouns, verbs, verb conjugation, etc.) must be hardwired into the human brain. They're not just taught to children by their parents: the children are "pre-configured" to learn grammar.

The argument says, suppose that weren't the case; suppose instead that babies started out as blank slates. Before it has to start speaking, a baby will hear, what, a hundred thousand sentences? Chomsky claims, with a bit of handwaving, that isn't nearly enough sentences to infer the general rules of grammar, the rules separating grammatical sentences from ungrammatical ones. The sample data available to the baby are just too impoverished for it to learn a language from scratch.

But here's the problem: the sample complexity bounds we saw earlier today should make us skeptical of any such argument! These bounds suggested that, in principle, it really *is* possible to predict the future given a surprisingly small amount of sample data. As long as the VC-dimension of your concept class is small—and I know I haven't convinced you of this, but in "most" practical cases it is—the amount of data needed for learning will be quite reasonable.

So the real stumbling block would seem to be not sample complexity, but computational complexity! In other words, if the basic ingredients of language weren't hardwired into every human baby, then even if in principle a baby has heard enough sentences spoken by its parents to infer the rules of the English language, how would the baby actually do the computation? It's just a baby!

More concretely, let's say I give you a list of n-bit strings, and I tell you that there's some nondeterministic finite automaton M, with much fewer than n states, such that each string was produced by following a path in M. Given that information, can you reconstruct M (probably and approximately)? It's been proven that if you can, then you can also break RSA! Now, finite automata are often considered the very simplest models of human languages. The grammar of any real human language is much too rich and complicated to be captured by a finite automaton. So this result is saying that even learning the least expressive, unrealistically simple languages is already as hard as breaking RSA!

So, using what we've learned about cryptography and computational learning theory, I submit

that we can now make the argument that Chomsky should have made but didn't. Namely: grammar must be hard-wired, since if a baby were able to pick up grammar from scratch, that baby could also break the RSA cryptosystem.[1]

2 Quantum Computing

Up to now in this course, you might have gotten the impression that we're just doing pure math—and in some sense, we are! But for most of us, the real motivation comes from the fact that computation is not just some intellectual abstraction: it's something that actually takes place in our laptops, our brains, our cell nuclei, and maybe all over the physical universe. So given any of the models we've been talking about in this course, you can ask: does this mesh with our best understanding of the laws of physics?

Consider the Turing machine or the circuit models. For at least a couple of decades, there was a serious question: will it be possible to build a general-purpose computer that will scale beyond a certain size? With vacuum tubes, the answer really wasn't obvious. Vacuum tubes fail so often that some people guessed there was a fundamental physical limit on how complex (say) a circuit or a Turing machine tape head could be before it would inevitably fail. In the 1950s, John von Neumann (who we met earlier) became interested in this question, and he proved a powerful theorem: it's possible to build a reliable computer out of unreliable components (e.g., noisy AND, OR, and NOT gates), provided the failure probability of each component is below some critical threshold, and provided the failures are uncorrelated with each other.

But who knows if those assumptions are satisfied in the physical universe? What really settled the question was the invention of the transistor in 1947—which depended on understanding semiconductors (like silicon and germanium), which in turn depended on the quantum revolution in physics eighty years ago. In that sense, every computer we use today is a quantum computer.

But you might say, this is all for the EE people. Once you've got the physical substrate, then we theorists can work out everything else sitting in our armchairs. But can we really?

Consider: what do we mean by efficiently solvable problem? Already in this course, you've seen two plausible definitions: P (the class of problems solvable by polytime deterministic algorithms) and BPP (the class solvable by polytime randomized algorithms with bounded error probability). The fact that we already had to change models once—from P to BPP—should make you suspicious, and this despite the fact that nowadays we conjecture that $P = BPP$. Could nature have *another* surprise in store for us, besides randomness?

[1] The ideas in this section are developed in much more detail in Ronald de Wolf's Masters thesis, "Philosophical Applications of Computational Learning Theory": http://homepages.cwi.nl/ rdewolf/publ/philosophy/phthesis.pdf