6.080 / 6.089 Great Ideas in Theoretical Computer Science
Spring 2008

6.080/6.089 GITCS 1 April 2008

Lecture 20

Lecturer: Scott Aaronson *Scribe: Geoffrey Thomas*

Probably Approximately Correct Learning

In the last lecture, we covered Valiant's model of "Probably Approximately Correct" (PAC) learning. This involves:

S:	A *sample space* (e.g., the set of all points)
D:	A *sample distribution* (a probability distribution over points in the sample space)
$c : S \to \{0, 1\}$:	A *concept*, which accepts or rejects each point in the sample space
C:	A *concept class*, or collection of concepts

For example, we can take our sample space to be the set of all points on the blackboard, our sample distribution to be uniform, and our concept class to have one concept corresponding to each line (where a point is accepted if it's above the line and rejected if it's below it). Given a set of points, as well as which points are accepted or rejected, our goal is to *output a hypothesis* that explains the data: e.g., draw a line that will correctly classify most of the future points.

A bit more formally, there's some "true concept" $c \in C$ that we're trying to learn. Given sample points $x_1, \ldots, x_m$, which are drawn independently from D, together with their classifications $c(x_1), \ldots, c(x_m)$, our goal is to find a hypothesis $h \in C$ such that

$$\mathbf{Pr}\left[h(x) = c(x)\right] \geq 1 - \epsilon$$

. Furthermore, we want to succeed at this goal with probability at least $1 - \delta$ over the choice of x_i's. In other words, with high probability we want to output a hypothesis that's approximately correct (hence "Probably Approximately Correct").

How many samples to learn a finite class?

The first question we can ask concerns *sample complexity*: how many samples do we need to have seen to learn a concept effectively? It's not hard to prove the following theorem: after we see

$$m = O\left(\frac{1}{\epsilon} \log \frac{|C|}{\delta}\right)$$

samples drawn from D, *any* hypothesis $h \in C$ we can find that agrees with all of these samples (i.e., such that $h(x_i) = c(x_i)$ for all i) will satisfy

$$\mathbf{Pr}\left[h(x) = c(x)\right] \geq 1 - \epsilon$$

with probability at least $1 - \delta$ over the choice of $x_1, \ldots, x_m$.

We can prove this theorem by the contrapositive. Let $h \in C$ be any "bad" hypothesis: that is, such that $\mathbf{Pr}\left[h(x) = c(x)\right] < 1 - \epsilon$. Then if we independently pick m points from the sample distribution D, the hypothesis h will be correct on all of these points with probability at most $(1 - \epsilon)^m$. So by the union bound, the probability that there *exists* a bad hypothesis in C that nevertheless agrees with all our sample data is at most $|C|\,(1 - \epsilon)^m$ (the number of hypotheses,

good or bad, times the maximum probability of each bad hypothesis agreeing with the sample data). Now we just do algebra:

$$\begin{aligned} \delta &= |C|\,(1-\epsilon)^m \\ m &= \log_{1-\epsilon} \frac{\delta}{|C|} \\ &= \frac{\log \delta/|C|}{\log 1-\epsilon} \\ &\approx \frac{1}{\epsilon} \log \frac{|C|}{\delta}. \end{aligned}$$

Note that there always *exists* a hypothesis in C that agrees with c on all the sample points: namely, c itself (i.e. the truth)! So as our learning algorithm, we can simply do the following:

1. Find any hypothesis in $h \in C$ that agrees with all the sample data (i.e., such that $h(x_i) = c(x_i)$ for all $x_1, \ldots, x_m$).
2. Output h.

Such an h will always exist, and by the theorem above it will probably be a good hypothesis. All we need is to see enough sample points.

How many samples to learn an infinite class?

The formula

$$m \approx \frac{1}{\epsilon} \log \frac{|C|}{\delta}$$

works so long as $|C|$ is finite, but it breaks down when $|C|$ is infinite. How can we formalize the intuition that the concept class of lines is learnable, but the concept class of arbitrary squiggles is not? A line seems easy to guess (at least approximately), if I give you a small number of random points and tell you whether each point is above or below the line. But if I tell you that *these* points are on one side of a squiggle, and *those* points are on the other side, then no matter how many points I give you, it seems impossible to predict which side the next point will be on.

So what's the difference between the two cases? It can't be the number of lines versus the number of squiggles, since they're both infinite (and be taken to have the same infinite cardinality).

From the floor: Isn't the difference just that you need two parameters to specify a line, but infinitely many parameters to specify a squiggle?

That's getting closer! The trouble is that the notion of a "parameter" doesn't occur anywhere in the theory; it's something we have to insert ourselves. To put it another way, it's possible to come up with silly parameterizations where even a line takes infinitely many parameters to specify, as well as clever parameterizations where a squiggle can be specified with just one parameter.

Well, the answer isn't obvious! The idea that finally answered the question is called *VC-dimension* (after two of its inventors, Vapnik and Chervonenkis). We say the set of points $x_1, \ldots, x_m$ is *shattered* by a concept class C if for all 2^m possible settings of $c(x_1), \ldots, c(x_m)$ to 0 or 1 (reject or accept), there is some concept $c \in C$ that agrees with those values. Then the VC-dimension of C, denoted $\mathrm{VCdim}(C)$, is the size of the largest set of points shattered by C. If we can find arbitrarily large (finite) sets of points that can be shattered, then $\mathrm{VCdim}(C) = \infty$.

If we let C be the concept class of lines in the plane, then $\text{VCdim}(C) = 3$. Why? Well, we can put three points in a triangle, and each of the eight possible classifications of those points can be realized by a single line. On the other hand, there's no set of four points such that all sixteen possible classifications of those points can be realized by a line. Either the points form a quadrilateral, in which case we can't make opposite corners have the same classification; or they form a triangle and an interior point, in which case we can't make the interior point have a different classification from the other three points; or three of the points are collinear, in which case we certainly can't classify those points with a line.

Blumer et al.[1] proved that a concept class is PAC-learnable if and only if its VC-dimension is finite, and that

$$m = O\left(\frac{\text{VCdim}(C)}{\epsilon}\log\frac{1}{\delta\epsilon}\right)$$

samples suffice. Once again, a learning algorithm that works is just to output *any* hypothesis h in the concept class that agrees with all the data. Unfortunately we don't have time to prove that here.

A useful intuition is provided by a corollary of Blumer et al.'s result called the *Occam's Razor Theorem*: whenever your hypothesis has sufficiently fewer bits of information than the original data, it will probably correctly predict most future data drawn from the same distribution.

Computational Complexity of Learning

We've seen that given a finite concept class—or even an infinite class with finite VC-dimension—after seeing enough sample points, you can predict the future just by finding any hypothesis in the class that fits the data. But how hard is it *as a computational problem* to find a hypothesis that fits the data? This has the general feel of something that might be NP-complete! In particular, it feels similar to satisfiability—find some hypothesis that satisfies certain fixed outputs—though it's not quite the same.

Here we need to make a subtle distinction. For *proper* learning—where the goal is to output a hypothesis in some fixed format (like a DNF expression), it's indeed possible to prove in some cases that finding a hypothesis that fits the data is NP-complete. For *improper* learning—where the hypothesis can be any polynomial-time algorithm so long as it predicts the data—to this day we don't know whether finding a hypothesis is NP-complete.

On the other hand, the learning problem is certainly *in* NP, since given a hypothesis it's easy to check whether it fits the data or not. This means that if $P = NP$, then all learning problems are in P and are computationally tractable. Think about what that means: we could ask our computer to find the shortest efficient description of the stock market, the patterns of neural firings in a human brain, etc., and thereby solve many of the hardest problems of AI! This is yet another reason to believe $P \neq NP$.

[1] Blumer, Ehrenfeucht, Haussler, Warmuth, "Learnability and the Vapnik-Chervonenkis dimension", JACM, 1989