COS 511: Foundations of Machine Learning

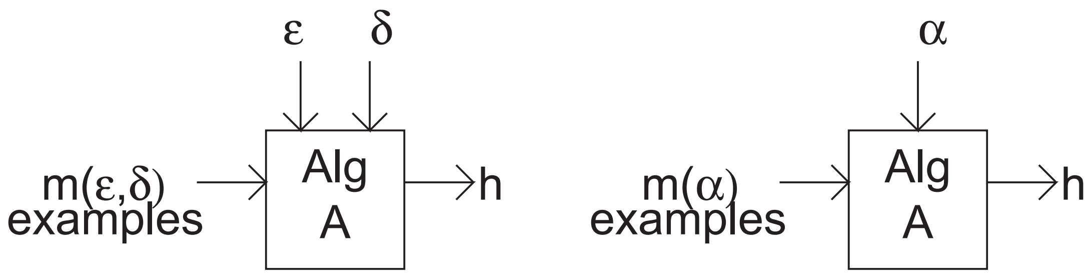
Rob Schapire — Lecture #4
Scribe: Jordan Boyd-Graber — February 16, 2006

1 Loose Ends

Before we address new material, some loose ends remain to be addressed from the last lecture.

1.1 Equivalence of PAC Learning Models

First, we'll address the issue of whether a learner with an expected error α, which we'll call an EC (expected correct) model, is equivalent to the PAC model that is defined in terms of a confidence δ and an accuracy ϵ.



We can imagine both of these learning models as black boxes that take as their input learning parameters (either α or ϵ and δ) and m examples (where m is a function of the learning parameters). Thus, we can form the question of equivalence as creating one learner using the inputs of the other.

In the first direction, given a PAC algorithm A, could an EC model use A as a subroutine to create a new learner that fulfills the EC constraints? Let's examine the expected error of A.

$$E[error] = Pr[error > \epsilon]E[error|error > \epsilon] + P[error \leq \epsilon]E[error|error \leq \epsilon]. \quad (1)$$

By the definition of the PAC learning model, $P[error > \epsilon]$ for a PAC learner is less than or equal to δ, and the expected value for the error if the error is known to be less than ϵ is less than ϵ. The other terms are no larger than one, so the expected value of the error is:

$$E[error] \leq \delta + \epsilon. \quad (2)$$

Thus, given some EC parameter of α, we run PAC using $\epsilon = \delta = \frac{\alpha}{2}$.

Going the other way, can we create a PAC learner with a given ϵ and δ given an EC algorithm? Again, let's look at the expected error. But this time we'll want to lower bound the error so that we can derive an expression for the probability of the error being greater than ϵ.

$$E[error] = Pr[error > \epsilon]E[error|error > \epsilon] + Pr[error \leq \epsilon]E[error|error \leq \epsilon]. \qquad (3)$$

We know that the error can be no lower than zero and that it's greater than ϵ for the first part of the sum; thus, the expected error is greater than $Pr[error > \epsilon]\epsilon$. Rearranging, this gives us:

$$Pr[error \geq \epsilon] \leq \frac{E[error]}{\epsilon} \leq \frac{\alpha}{\epsilon} \leq \delta. \qquad (4)$$

(The first inequality holds for any non-negative random variable and is known as Markov's inequality.) So we want to run the EC algorithm with α set to $\epsilon\delta$.

1.2 Must the target concept class be in the hypothesis space?

In general, it need not be the case that the target concept class C is contained in the hypothesis class H, at least when the domain X is infinite. For instance, consider the case when $X = \mathbb{R}$, and the target class C consists of all positive half-lines. If the hypothesis space H were only those positive half-lines that began on rational numbers, then although we have only a measure-zero fraction of the possible concepts, we can come arbitrarily close to any concept. In terms of learning, we have already seen that any consistent hypothesis will be PAC, and in particular, we can choose an algorithm which, given a finite sample, selects a rational separating point (and thus, a member of H) that falls between the sampled points.

On the other hand, when X is finite, it can be argued that C must be in H. Suppose $k = |X|$, and pick any $c \in C$. Because the PAC learner must work for all distributions D, error ϵ, and confidence δ, we choose D such that it is uniform over X, $\epsilon = \frac{1}{2k}$, and $\delta = \frac{1}{2}$. The PAC criterion tells us that it is possible to find a hypothesis h whose error is less than or equal to $\frac{1}{2k}$, which is less than $\frac{1}{k}$, the probability of a single point under distribution D. So when we run A (on a possibly very large data set), it gives us an h that agrees with c on *all* points in X. In other words, c and h must be identical functions. Thus, $c \in H$, and therefore $C \subseteq H$.

1.3 Bogosity of rectangle argument

The proof technique discussed in the last lecture for PAC learnability of points in $\mathbb{R}^2$ only applies to the *smallest* consistent rectangle.

2 A generalized argument for PAC learnability

Theorem 1 *If A finds $h_A \in H$ consistent with m examples where $m \geq \frac{1}{\epsilon}(\ln|H| + \ln\frac{1}{\delta})$, then $Pr[h_A$ ϵ-bad$] \leq \delta$.*

This is a very general bound on the sample complexity, and has a fairly simple proof. We want to bound the probability of h_A being consistent *and* being ϵ-bad.

But all we know is that $h_A \in H$; we know nothing about how it was chosen or its properties. We want to form a bound based on the probability of some hypothesis that both consistent and ϵ-bad.

Because we don't know anything about how h is chosen, we can merely make a statement about the probability of such an h existing.

Let's now define a set B that contains all hypotheses that are ϵ-bad. This allows us to write the expression in terms of a conjunction over the elements of B.

$$\begin{aligned} Pr[h_A \text{ cons} \wedge h_A \ \epsilon\text{-bad}] &\leq Pr[\exists h \in H : h \text{ cons} \wedge \epsilon\text{-bad}] \\ &= Pr[\exists h \in B : h \text{ cons}] \\ &= Pr\left[\bigvee_{h \in B} h \text{ cons}\right]. \end{aligned}$$

Applying the union bound, we get that the probability is:

$$Pr[\exists h \in B : h \text{ cons}] \leq \sum_{h \in B} Pr[h \text{ cons}]. \tag{5}$$

Now the question turns to determing the probability that h is consistent given that it is ϵ-bad for some ϵ. Remember, for our sample $S = \langle (x_1, c(x_1)), \ldots (x_m, c(x_m)) \rangle$, h must always agree with c. Thus, we use the independence of h agreeing on any given example to get the following:

$$\begin{aligned} Pr[h \text{ cons}] &= Pr[h(x_1) = c(x_1) \wedge \ldots \wedge h(x_m) = c(x_m)] \\ &= \prod_{i=1}^{m} Pr[h(x_i) = c(x_i)]. \end{aligned}$$

But for any individual x_i the probability of them matching up must be less than or equal to $(1 - \epsilon)$ because $h \in B$. Thus, over m examples, the probability of h being consistent is at most $(1 - \epsilon)^m$. So when we sum over all of the $|B|$ hypotheses and use the fact that $B \subseteq H$, we get the following:

$$Pr[\exists h \in B : h \text{ cons}] \leq \sum_{h \in B} Pr[h \text{ cons}] \leq |B|(1 - \epsilon)^m \leq |H|(1 - \epsilon)^m. \tag{6}$$

Applying the inequality that $(1 - x) \leq e^{-x}$ and substituting in m as we defined it earlier, we have:

$$Pr[\exists h \in B : h \text{ cons}] = |H| e^{-\epsilon m} \leq \delta. \tag{7}$$

Which is what we wanted to prove. Notice that $\ln |H|$ acts like a complexity term that is proportional to $\lg |H|$, the number of bits needed to represent a hypothesis in binary. For example, monotone conjunctions require n bits over n variables (each term is either present or not), so m grows as $n \ln 2$.

Turning it around, the error of a hypothesis is less than $\frac{1}{m}(\ln |H| + \ln \frac{1}{\delta})$.

3 An alternative approach to the error

Suppose that we want another bound on the error independent of the structure of H. Suppose instead that we want to only compute the probability of one particular h_A being ϵ-bad (i.e. the hypothesis that was the result of our learning procedure). We now use the fact that the denominator is 1 since we are only concerned with a consistent hypothesis:

$$Pr[h_A\ \epsilon\text{-bad}|h_A\ \text{cons}] = \frac{Pr[h_A\ \epsilon\text{-bad} \wedge h_A\ \text{cons}]}{Pr[h_A\ \text{cons}]} = Pr[h_A\ \text{cons}|h_A\ \epsilon\text{-bad}]Pr[h_A\ \epsilon\text{-bad}]. \tag{8}$$

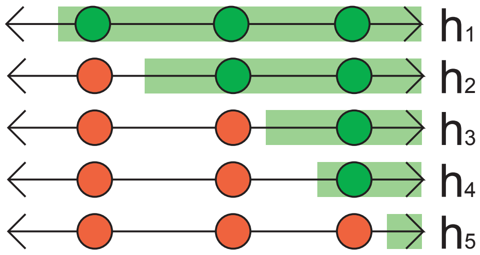
Now, at this point we might want to use our previous result that gave us:

$$Pr[h_A\ \text{cons}|h_A\ \epsilon\text{-bad}] = \prod_{i=1}^{m} Pr[h_A(x_i) = c(x_i)|h_A\ \epsilon\text{-bad}]. \tag{9}$$

But h_A depends on all of the elements of S, so this doesn't hold because these events are no longer independent and the individual probabilities are no longer less than $1-\epsilon$. So the moral of this lesson is that these arguments can be slippery. We especially need to be careful, as we were in our earlier argument, to identify the hypothesis whose error we want to analyze *before* the sample is selected.

4 Learnability of Infinite Hypothesis Spaces

The result above only holds for finite hypothesis spaces. There's something, though, about half-lines, rectangles, etc. that still allow us to learn how to classify them even though they're infinite. So let's look at $X = \mathbb{R}$ and $H = C = \{$ positive half-lines $\}$. In the picture below, we have m examples and we form a number of hypotheses. Let's see how they categorize the points into positive (green) and negative (red).



So even though h_3 and h_4 are different hypotheses, for the data that we have, they have the same behavior for m points. In fact, there are only $m+1$ "slots" where we can put our positive half-lines, so there are only $m+1$ behaviors or dichotomies for this class of functions.

If we consider all possible dichotomies on m points, then there are 2^m possible ways of assigning a classification to m points. "Nice" cases will be polynomial in m, while the total size of the possible dichotomies will be exponential. We can define the set of all dichotomies for a sample $S = \langle x1, \ldots, x_m \rangle$ for a particular hypothesis space H:

$$\Pi_H(S) = \{\langle h(x_1), \ldots, h(x_m) \rangle : h \in H\}. \tag{10}$$

We can then define a growth function over m samples to capture how complex our hypothesis space grows as we see more samples.

$$\Pi_H(m) = max_{|S|=m}|\Pi_H(S)|. \tag{11}$$

Intuitively, we want this measure to replace $|H|$ term in our error bound so that the complexity measure $\ln |H|$ becomes $\ln |\Pi_H(m)|$. This means that if the growth function grows as 2^m, then we get a meaningless bound of $m \geq \frac{m+\ln \frac{1}{\delta}}{\epsilon}$. In this case, and as we will see, learning is impossible because we are working with something like all possible functions.

However, most of the interesting cases are $O(m^d)$, where d is the VC-dimension. Next time we'll discuss the beautiful result that tells us:

Theorem 2 *For any H, either $\Pi_H(m) = 2^m$ for all m or $\Pi_H(m) = O(m^d)$.*