COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Yuhui Luo

Lecture #3
February 11, 2008

1 The Probably Approximately Correct (PAC) Model

A target concept class $\mathcal{C}$ is PAC-learnable by $\mathcal{H}$ (the hypothesis class) if there exists an algorithm A which for all concepts $c \in \mathcal{C}$, all distributions D, and all $\epsilon > 0, \delta > 0$ takes $m = poly(1/\epsilon, 1/\delta, \ldots)$ examples defined by the training set $\mathcal{S} = \langle (x_1, c(x_1)), \cdots, (x_m, c(x_m)) \rangle$ and produces a hypothesis $h \in \mathcal{H}$ such that $\Pr[err_D(h) \leq \epsilon] \geq 1 - \delta$ (the hypothesis is ϵ-good).

In this model, ϵ is known as the error parameter, δ is known as the confidence parameter. $err_D(h) = \Pr[h(x) \neq c(x)]$ is known as the true error or generalization error. And $\frac{1}{m} |\{i : h(x_i) \neq c(x_i)\}|$ is the training or empirical error. We also assume that every x_i from the training set and all the test points are drawn from the same target distribution D.

The benefit of a PAC learning model is that it actually makes a real connection to learning. It captures what learning is: taking a set of examples and generalizing it. It can learn to make predictions accurately.

2 A Simple Learning Problem

As an example, let's look at a simple learning problem. Our domain, $\mathcal{X}$ is the real line. Our concept class, $\mathcal{C}$ is the set of all positive half-lines, which are lines that separate the real line into two halves: all the points to the left of the line are negatively labelled and all the points to the right of the half-line are positively labelled.

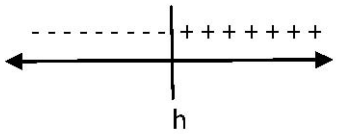

Figure 1: A Simple Learning Problem

Given a set of labelled points, a valid hypothesis could be to find the place where the examples transition from negative labels to positive labels, and place the half-line anywhere in the transition region (see Figure 1). For example, we can place the half-line midway between the right-most negative and left-most positive examples.

Is this a PAC learning algorithm? In particular, if it is a PAC algorithm, how many examples do we need in order to ensure that this learning algorithm will be ϵ-good with high probability?

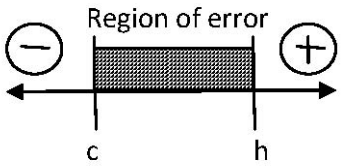

Figure 2: The Region of Error

As seen in Figure 2, the region of error is the region between c and h: If we get any points there, h will say it's negative while c would say it's positive. We can also have c to the right of h, where a similar argument can be made due to symmetry. Now, we want the error region to be at most ϵ. More precisely, we want the probability of a point from our target distribution D falling into this region to be less than ϵ. So, when we talk about distances, we're not talking about the actual distance, but the distance with respect to the probability distribution.

Now, let's go back to the time before any data points are chosen (see Figure 3).

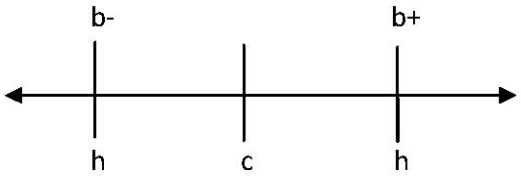

Figure 3: The Two Bad Events

We know that there exists some target concept c. And we know that there are two bad events:

1. **b+**: The hypothesis h is more than ϵ to the right of the target concept c.
2. **b-**: The hypothesis h is more than ϵ to the left of the target concept c.

As long as those two events don't happen with high probability, then we can say that h is ϵ-good. Since these two events are symmetric, we only need to look at one of them at a time.

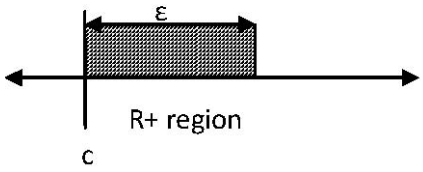

Figure 4: Region of Probability ϵ

Figure 4 shows the **b+** event and a region of probability ϵ to the right of c, called $R+$. In this region, the probability mass between c and the edge of the region is ϵ. We want to bound the probability of h ending up outside the region R^+. If there's even a single point

that falls in R^+ – a positive example, since it falls to the right of c – then h will fall to the left of that point and also be inside the region R^+. Therefore, the only way for the bad event to happen is if no points fall inside the R^+ region. That is:

$$\begin{aligned}\Pr[\mathbf{b}+] &\leq \Pr[x_1 \notin R^+ \wedge \cdots \wedge x_m \notin R^+] \\ &= \Pr[x_1 \notin R^+] \cdots \Pr[x_m \notin R^+] \\ &= (1-\epsilon)^m \\ &\leq e^{-\epsilon m}.\end{aligned}$$

Here, we use the assumption that all x_i are i.i.d (independent and identically distributed), where the probability of x_i falling in the region R^+ is ϵ. We also make use of the inequality $1 + x \leq e^x$ to obtain the last line of the result. Then:

$$\begin{aligned}\Pr[h\ \epsilon\text{-bad}] &\leq \Pr[\mathbf{b}+ \vee \mathbf{b}-] \\ &\leq \Pr[\mathbf{b}+] + \Pr[\mathbf{b}-] \\ &\leq 2e^{-\epsilon m}\end{aligned}$$

where the second line was obtained using the union bound and the third line was obtained by symmetry. We want:

$$\Pr[h\ \epsilon\text{-bad}] \leq \delta.$$

Solving for m, we obtain:

$$m \geq \frac{1}{\epsilon} \ln \frac{2}{\delta}.$$

Therefore, if we have at least $m \geq \frac{1}{\epsilon} \ln \frac{2}{\delta}$ examples, our algorithm is PAC-learnable. Solving for ϵ, we can also say that with probability $1 - \delta$, $err_D(h) \leq \frac{1}{m} \ln \frac{2}{\delta}$.

3 More Learning Problems

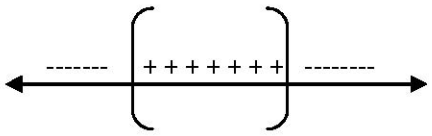

Figure 5: Interval Learning Problem

Now suppose we keep the same domain as the previous problem, but change the concept class to be intervals instead, where each interval contains all the positively labelled examples and all points outside the interval are considered negative (see Figure 5).

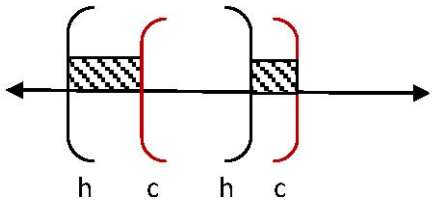

Figure 6: Interval Learning Problem

In this case, for the entire region of error to be less than ϵ, we can specify that the two separate regions of error in Figure 6 each be less than $\epsilon/2$. Now we can use the exact same argument as our previous problem, though instead of two bad events, we now have four bad events. This yields a bound of $m \geq \frac{2}{\epsilon} \ln \frac{4}{\delta}$.

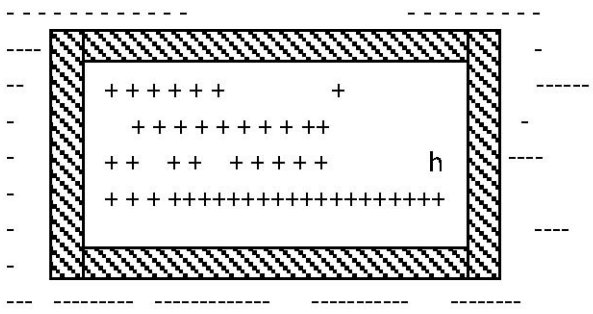

Figure 7: A Learning Problem in $\mathbb{R}^2$

Now suppose we set our domain $\mathcal{X} = \mathbb{R}^2$, and set our concept clas $\mathcal{C} = \{$axis-parallel rectangles$\}$. Suppose we use the same algorithm from our previous lecture: we take the smallest rectangle that contains all the positive points and set that as our hypothesis. Since it's the smallest rectangle, we know that it must be contained inside the concept c. Then, our regions of error are shown in Figure 7 and we want each to have an error less than $\epsilon/4$. Once again, we can then follow the same arguments as our previous learning problems to derive a bound.

We can also extend this problem to $\mathbb{R}^n$, where our concept class then becomes all axis-parallel hyper-rectangles. In these problems, it is common to have a size parameter of the concept class, n. Then, we will allow the number of examples to be polynomial in n as well as the usual $1/\epsilon$ and $1/\delta$.

4 Expected Correct (EC) Learning Model

A different learning model is the EC learning model, which instead of restricting the probability of error being greater than ϵ, restricts the expected value of the error.

In this model, a target concept class $\mathcal{C}$ is EC-learnable by $\mathcal{H}$ (the hypothesis class) if there exists an algorithm A which for all concepts $c \in \mathcal{C}$, all distributions D, and all $\alpha > 0$ takes $m = poly(1/\alpha, \ldots)$ examples defined by the training set $\mathcal{S} = \langle (x_1, c(x_1)), \cdots, (x_m, c(x_m)) \rangle$ and produces a hypothesis $h \in \mathcal{H}$ such that $\mathrm{E}[err_D(h)] \leq \alpha$.

Now, the question is: are the PAC and EC models equivalent? For two models to be equivalent, anything which can be learned in one model must also be learnable in the other model. So, if we are handed a PAC learning algorithm, then we should be able to use it to create an EC learning algorithm and vice versa.

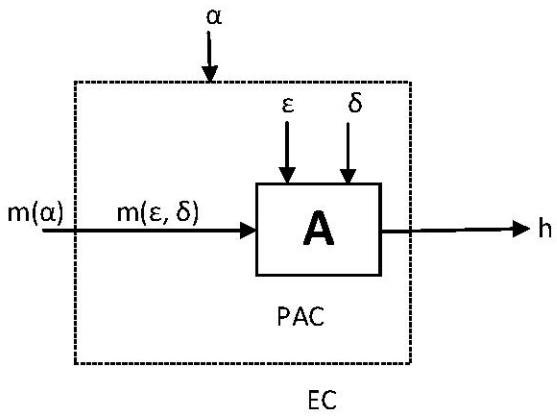

Figure 8: PAC to EC Model

In Figure 8, we are given a PAC model and we want to see if we can convert it to an EC learning model. We can calculate the expected value of the error as follows:

$$\begin{aligned} \mathrm{E}[err] &= \Pr[err > \epsilon]\mathrm{E}[err|err > \epsilon] + \Pr[err \leq \epsilon]\mathrm{E}[err|err \leq \epsilon] \\ &\leq \delta + \epsilon. \end{aligned}$$

where we use the fact that $\Pr[err > \epsilon] \leq \delta$, $\mathrm{E}[err|err > \epsilon] \leq 1$, $\Pr[err \leq \epsilon] \leq 1$ and $\mathrm{E}[err|err \leq \epsilon] \leq \epsilon$ to obtain the final expression. Therefore, we can choose $\epsilon = \delta = \alpha/2$ to obtain an EC learning model given a PAC learning model. A similar argument can be made for the reverse.

5 A General Result

Back to the PAC learning model, we now want to know if there is a more general result for showing that an algorithm is PAC-learnable. Can we use consistency to come up with a general result on the number of examples we need?

Theorem. *Suppose we are given a finite hypothesis space (i.e. $|\mathcal{H}| < \infty$). Say an algorithm A finds a hypothesis $h_A \in \mathcal{H}$ consistent with m examples, where $m \geq \frac{1}{\epsilon}\left(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\right)$. Then, the hypothesis is ϵ-good. That is:*

$$\Pr[err_D(h_A) > \epsilon] \leq \delta.$$

5.1 Dependence on $|\mathcal{H}|$

Why does this general bound have a dependence on the size of $\mathcal{H}$? Intuitively, we can say that the more rules there are in the hypothesis space, the more opportunities there are for us to latch onto a bad rule that works well for the training samples, but does not work well for predicting the test cases.

Exercise: Everyone in the class writes down a 0 or a 1 for 10 training points and for 10 test points. This is matched against 20 random coin tosses Professor Schapire did before class. The highest accuracy guess had a training error of 20%. This matches well with our intuition that given many hypotheses, we will find a hypothesis that works very well with the training data. However, the person who made the best guess for the training data ended up with a test error of 60% when it came time to match the guesses against the test data.

5.2 Why is the Bound Logarithmic?

For the bound to be logarithmic, it essentially captures the complexity of the hypothesis space. Let's say we gave each rule a name (in binary), how many bits do we need?

Answer: $\log_2 |\mathcal{H}|$.

Example 1: Suppose our hypothesis class consists of all monotone conjunctions. Here, $\mathcal{X} = \{0,1\}^n$ and $|\mathcal{H}| = 2^n$. Therefore, $m \geq \frac{1}{\epsilon}(n \ln 2 + \ln 1/\delta)$ provides an immediate bound on the data we needed for the algorithm earlier presented for this class.

Example 2: Now suppose our hypothesis class, $\mathcal{H} = \{DNF\}$. Then, $|\mathcal{H}| = 2^{2^n}$. Therefore, $m \geq \frac{1}{\epsilon}(2^n \ln 2 + \ln 1/\delta)$ and the bound is very poor (exponential) validating our intuition that DNF is too rich a class to learn efficiently with a small amount of data.