COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Kevin Lai

Lecture #3
February 11, 2014

1 The Probably Approximately Correct (PAC) Model

A target concept class $\mathcal{C}$ is **PAC-learnable** by a hypothesis space $\mathcal{H}$ if there exists an algorithm A such that for all $c \in \mathcal{C}$, any target distribution D, and any positive ϵ and δ, A uses a training set $S = \langle (x_1, c(x_1)), (x_2, c(x_2)), ..., (x_m, c(x_m)) \rangle$ consisting of $m = \text{poly}(\frac{1}{\epsilon}, \frac{1}{\delta}, ...)$ examples taken i.i.d. from D and produces $h \in \mathcal{H}$ such that $\Pr[\text{err}_D(h) \le \epsilon] \ge 1 - \delta$.

A few comments on notation. ϵ is called the accuracy parameter, and we call h "ϵ-good" if $\text{err}_D(h) \le \epsilon$, where $\text{err}_D(h)$ is called the **true error** or the **generalization error**. δ is the confidence parameter. ϵ and δ are user-specified parameters (eg. 5% and 1%). The name "Probably Approximately Correct" comes from the fact that we want a hypothesis that is approximately correct (ϵ-good) with high probability (namely $1-\delta$). The probability is taken over the choice of S, which will determine which h the algorithm chooses. This is a reasonable goal because there is always a small chance that the test data will be very unrepresentative of D.

We assume that the training set and the test data are drawn from the same distribution D. In general $\mathcal{H}$ will not necessarily be the same as $\mathcal{C}$. Finally, we may also want m to be polynomial in the size of each example or in the size of the target concept c.

2 Learning positive half-lines

We will now look at a series of examples of PAC-learnable concept classes, starting with the class of positive half-lines. In this example, the domain $\mathcal{X}$ is the real line, and $\mathcal{C} = \mathcal{H} = \{\text{positive half lines}\}$. A positive half-line is defined by a threshold (a real number): all points to the left of the threshold are labeled negative, while all points to the right of the threshold are labeled positive.

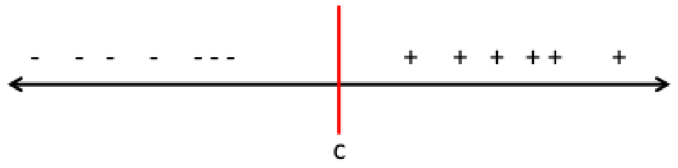

Figure 1: The target concept c is a half-line

To find a hypothesis, we will simply pick some h that is consistent with the test data. We can do this by scanning the test data in ascending sorted order until we find the greatest negatively labeled point and the smallest positively labeled point. We then set the threshold of our half-line anywhere in this interval. Note that we can always find a consistent h because $\mathcal{H} = \mathcal{C}$.

As shown in Figure 3, the generalization error of h will be the probability mass that falls between the target concept c and our hypothesis h. Points in this region will be labeled

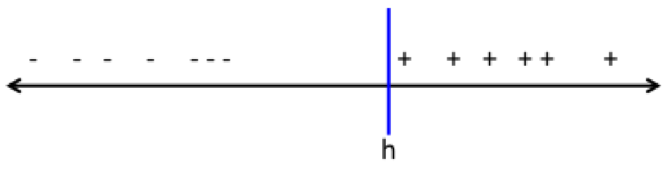

Figure 2: Half-line h outputted by our algorithm based on test data

differently by h and c. Any points that are either to the right or to the left of both h and c will be labeled the same by h and c.

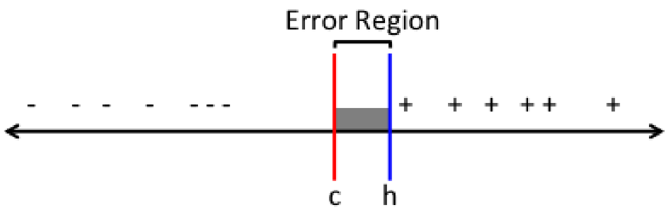

Figure 3: The generalization error of h is the probability mass between h and c

We need to show that the generalization error is low with high probability. Equivalently, we can show that $\Pr[\mathrm{err}_D(h) > \epsilon] \leq \delta$. There are two cases where $\mathrm{err}_D(h) > \epsilon$. Let B_+ be the event that h is more than ϵ probability mass to the right of c, and let B_- be the event that h is more than ϵ probability mass to the left of c.

To determine the probability that B_+ or B_- occur, we define two points on the number line, as shown in Figure 4. Let r_+ be a point to the right of c such that the interval $[c, r_+]$ has ϵ probability mass. Likewise, let r_- be a point to the left of c such that the interval $[r_-, c]$ has ϵ probability mass.

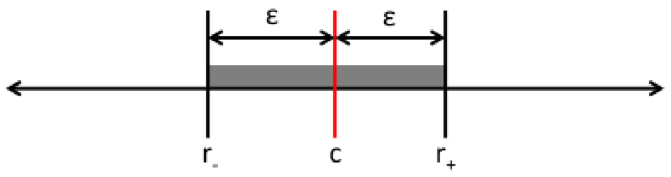

Figure 4: r_+ and r_- are both ϵ probability mass away from c

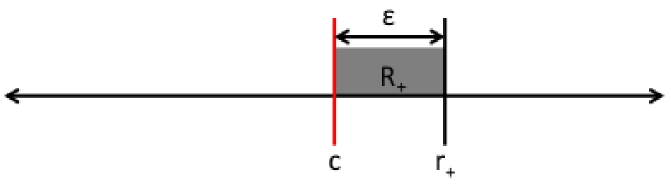

Figure 5: R_+ has probability mass ϵ

We will first calculate the probability of B_+. Let R_+ be the interval $[c, r_+]$, which has probability mass ϵ. Now note that B_+ can only occur if h is to the right of r_+, which only occurs if all $x_i \notin R_+$. If any of the training points were in R_+, h would be to the left of r_+

(since the training point would have a positive label). Observe now that $\Pr[x_i \notin R_+] \leq 1-\epsilon$ because R_+ has probability mass ϵ. Then:

$$\Pr[B_+] = \Pr[x_1 \notin R_+ \wedge x_2 \notin R_+ \wedge x_m \notin R_+] \leq (1-\epsilon)^m \tag{1}$$

where the last inequality follows by independence of the x_i's. Note also that while the x_i's are random, R_+ is fixed because it depends on a fixed c. Also, by symmetry, $\Pr[B_-] \leq (1-\epsilon)^m$.

Now we can bound the probability that $\text{err}_D(h) > \epsilon$:

$$\begin{aligned}
\Pr[\text{err}_D(h) > \epsilon] &\leq \Pr[B_+ \vee B_-] && (2)\\
&\leq \Pr[B_+] + \Pr[B_-] \qquad \text{(union bound)} && (3)\\
&\leq 2(1-\epsilon)^m && (4)\\
&\leq 2e^{-\epsilon m} \qquad (\forall x, 1-x \leq e^{-x}) && (5)
\end{aligned}$$

We want $\Pr[\text{err}_D(h) > \epsilon] \leq \delta$, so we set $(5) \leq \delta$ and solve for m to get $m \geq \frac{1}{\epsilon}\ln\frac{2}{\delta}$. This shows that $\mathcal{C}$ is PAC-learnable by $\mathcal{H}$.

If we set $\delta = (5)$, we can write this equivalent statement: with probability at least $1-\delta, \text{err}_D(h) \leq \frac{1}{m}\ln\frac{2}{\delta}$.

3 Learning intervals

Our next concept class $\mathcal{C}$ is the set of intervals on the real line. This will also be our hypothesis space $\mathcal{H}$. Each $c \in \mathcal{C}$ specifies an interval on the real line that will be labeled positive. All points outside of the interval are labeled negative.

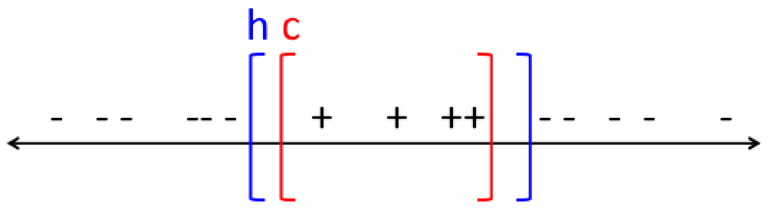

Figure 6: Test data generated by a target concept interval c. h is a consistent hypothesis interval

Our algorithm will simply pick a consistent $h \in \mathcal{H}$. We can then prove that the generalization error of h is low with high probability by an analogous argument to the half-line case. That is, $\mathcal{C}$ is PAC-learnable by $\mathcal{H}$.

We will briefly summarize the argument. Let c_l be the left boundary of c. We make an interval of probability mass $\frac{\epsilon}{2}$ on both sides of c_l. Then the analysis will be the same as in the half-line case, except with a constant factor difference. We repeat this argument for the right boundary of c. At the end, we will have a union bound over four bad events, each with probability $e^{-\frac{\epsilon m}{2}}$, so $m \geq \frac{2}{\epsilon}\ln\frac{4}{\delta}$.

4 Learning axis-aligned rectangles

Our last example will be learning the concept class $\mathcal{C}$ of axis-aligned rectangles. We will use the algorithm we had in the previous lecture, which finds the smallest axis-aligned rectangle consistent with the data.

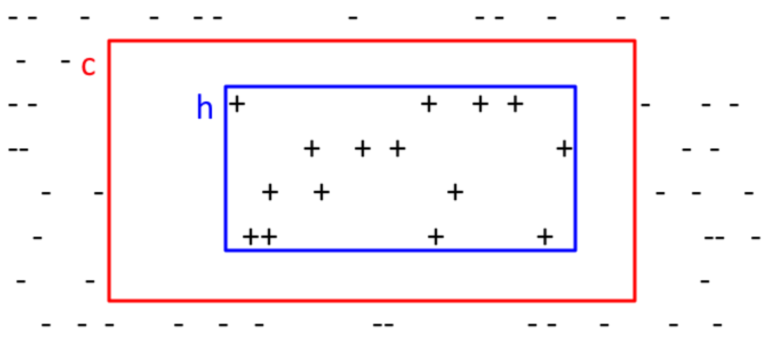

Figure 7: Test data generated by a target concept rectangle c. h is the smallest consistent rectangle

To bound the probability that the generalization error is greater than ϵ, we create four "bad" regions, each created by growing a rectangle from the interior of one wall of c until the region has probability mass $\frac{\epsilon}{4}$.[1] It does not matter that these rectangles overlap. Call these regions $R_1, ..., R_4$. Next, we use an argument that is analogous to the half-line case to bound the probability that no points land in each region individually. Let $B_1, ..., B_4$ be the events that no points land in regions $R_1, ..., R_4$ respectively. Then:

$$\begin{aligned}
\Pr[\mathrm{err}_D(h) > \epsilon] &\leq \Pr[B_1 \vee B_2 \vee B_3 \vee B_4] && (6)\\
&\leq \Pr[B_1] + \Pr[B_2] + \Pr[B_3] + \Pr[B_4] \qquad \text{(union bound)} && (7)\\
&\leq 4(1-\epsilon/4)^m && (8)\\
&\leq 4e^{-\frac{\epsilon m}{4}} && (9)
\end{aligned}$$

Then we get $m \geq \frac{4}{\epsilon} \ln \frac{4}{\delta}$. As a final note, observe that we can extend this argument to axis-aligned hyperrectangles in an arbitrary number of dimensions.

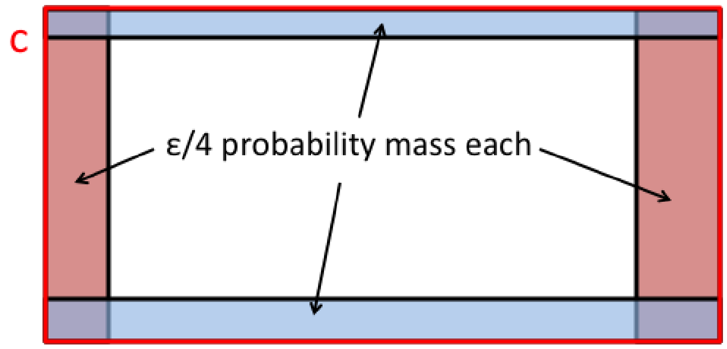

Figure 8: Regions analogous to R_+ in the half-line case. We bound the probability that no points fall in each of these regions. Each region has probability mass $\frac{\epsilon}{4}$ and the regions overlap

[1] If the region hits the opposite wall of c before the region reaches probability mass $\frac{\epsilon}{4}$, then we simply stop growing the region. Thus, each region has probability mass at most $\frac{\epsilon}{4}$.

5 Proving PAC results in general

We just saw several examples of proving concept classes are PAC-learnable using proofs tailored to each problem. However, it would be more convenient if we had some way to prove PAC results in general. Fortunately, such a way exists for finite hypothesis spaces (i.e. $|\mathcal{H}| < \infty$).

Theorem 1. *Suppose an algorithm A always finds a hypothesis $h_A \in \mathcal{H}$ consistent with m examples where $m \geq \frac{1}{\epsilon}(\ln|\mathcal{H}| + \ln\frac{1}{\delta})$. Then $\Pr[\text{err}_D(h_A) > \epsilon] \leq \delta$.*

The proof of Theorem 1 also will lead to this similar statement: with probability at least $1 - \delta$, if h_A is consistent, then $\text{err}_D(h_A) \leq \frac{1}{m}(\ln|\mathcal{H}| + \ln\frac{1}{\delta})$. This statement is essentially saying that if an algorithm finds a consistent hypothesis and uses enough data relative to the complexity of the hypothesis space and δ, the generalization error will be low.

6 Complexity and $|\mathcal{H}|$

The $\ln|\mathcal{H}|$ term in Theorem 1 is a measure of complexity of the hypothesis space. Intuitively, we can understand the logarithmic term as essentially the number of bits required to uniquely label each $h \in \mathcal{H}$, which would be the base two logarithm of $|\mathcal{H}|$.

We did an exercise in class to demonstrate the relevance of $|\mathcal{H}|$. Everyone wrote down 20 bits of their choosing, corresponding to the output of a hypothesis on the domain $\{1, 2, ..., 20\}$. Professor Schapire then revealed the "training" data, which was a set of 10 bits, corresponding to the output of the target concept for {1,2,...,10}. We selected the hypothesis with the lowest training error in the class. Then we evaluated that hypothesis's performance against the 10 bits output by the target concept for {11,12,...,20}. In our example, the best hypothesis had a training error of 10% and a test error of 10%. However, as Professor Schapire determined his bits using random coin tosses, in general the hypothesis with the best training error will have an expected test error of 50%. In a large class, it is likely to find a hypothesis with low training error, but that hypothesis will still have an expected test error of 50%.

The purpose of the exercise was to demonstrate that with a bigger hypothesis space, there is a greater chance for a poor hypothesis to match the training set well, despite having poor generalization error. Thus, complexity in the hypothesis space will tend to increase the probability of choosing a hypothesis that fits the training set well, simply by chance, but which actually performs poorly on test data.

Example 1. Suppose $\mathcal{C}$ is the set of monotone conjunctions in n-dimensions. This will also be our hypothesis space. $|\mathcal{H}| = 2^n$ because each variable can either be included or not included in our hypothesis. Since our algorithm from last lecture finds a consistent hypothesis, Theorem 1 implies that if $m \geq \frac{1}{\epsilon}(n \ln 2 + \ln\frac{1}{\delta})$, then $\Pr[\text{err}_D(h_A) > \epsilon] \leq \delta$. Since m only needs to be at most polynomial in the size of $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, and n, we have shown that the class of monotone conjunctions is PAC-learnable.

Example 2. Now suppose that $\mathcal{C}$ is the set of all DNFs, and let this be our hypothesis space as well. Now since we can think of every DNF as a boolean formula (given values for n variables, the DNF outputs one of two answers), and every boolean formula can be represented by a DNF, $|\mathcal{H}|$ is just the number of boolean formulas on n variables, which is 2^{2^n}. Using Theorem 1, we get that $m \geq \frac{1}{\epsilon}(\ln 2^{2^n} + \ln \frac{1}{\delta}) = \frac{1}{\epsilon}(2^n \ln 2 + \ln \frac{1}{\delta})$, which is exponential in the number of variables. This is consistent with our conclusions from last lecture that DNFs likely can't be learned efficiently with a small amount of data.

An informal way to see that the class of DNFs likely can't be learned efficiently is to look at our algorithm from last lecture for learning DNFs. In that algorithm, we included a clause for each of the positive examples in our training set. In the worst case, this could give us a hypothesis of size $O(mn)$. If we approximate $\ln |\mathcal{H}|$ with the size of each hypothesis, then the error is $\epsilon \geq \frac{O(mn)+\ln \frac{1}{\delta}}{m} \geq cn$, where c is some constant. This is too large, as we would like our error to be less than 1. So our previous algorithm does not efficiently learn DNFs.

In general, we see that if our hypothesis space is not too complex, finding a consistent hypothesis will allow us to achieve low generalization error with a reasonably small amount of data. However, this conclusion does not say anything about computational efficiency. Even if only a small amount of data is required, it may be difficult to design efficient learning algorithms.