COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Christian Eubank

Lecture 3
February 12, 2013

1 The Probably Approximately Correct (PAC) Model

1.1 Definition

The concept class $\mathcal{C}$ is PAC-learnable by the hypothesis class $\mathcal{H}$ if there exists an algorithm A such that for all $c \in \mathcal{C}$, for all distributions D, for all $\epsilon > 0$, for all $\forall \delta > 0$, A takes $m =$ polynomial number of examples given in the form $S = \langle (x_1, c(x_1)), ...(x_m, c(x_m)) \rangle$, where each x_i is chosen from the space X at random according to the target distribution D, and produces a hypothesis $h \in \mathcal{H}$ such that $\Pr[\text{err}_D(h) \leq \epsilon] \geq 1 - \delta$.

PAC learning meets our goal of using training data to construct a hypothesis that is also accurate on other yet-unseen data. However, this result is not entirely intuitive. After all, training examples are drawn from a huge universe and we claim that we can use a merely polynomially-sized sample to get high accuracy with high probability on future data.

1.2 Example: Learning on a line

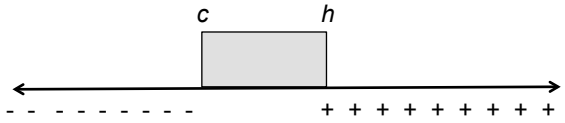

Figure 1: Hypothesis and error region for guessing half lines

Let $X = \mathbb{R}$ and $\mathcal{C} = \{\text{positive half lines}\}$. For some point c, the corresponding positive half line is the region of $\mathbb{R}$ designated by $[c, \infty)$. We can treat $c \in \mathcal{C}$ as a point that separates the positive and negative regions of $\mathbb{R}$. A natural hypothesis is to simply choose a point h that is consistent with the training data. As seen in Figure 1, there will be a region between c and h in which the hypothesis will incorrectly label new training points. We want this region to have size $\leq \epsilon$. Here, size does not refer to the actual size of the region in terms of length. Instead, we want to guarantee that, given the true distribution D of all possible points, we want no more than ϵ of these points to fall between c and h. In this model, we have two bad cases:

1. B^+ : h lies more than ϵ to the right of c
2. B^- : h lies more than ϵ to the left of c

Mathematically, we can reason about $\text{err}_D(h)$ by first considering the likelihood of B^+. As seen in Figure 2, let R^+ be the be the smallest region with c as its left border whose probability mass is at least ϵ. That is $R^+ = [c, r^+]$ where $r^+ = \sup\{r \geq c | \Pr[[c, r)] < \epsilon\}$. By our algorithm, h falls to the right of r^+ only if all training examples lie outside of R^+. Hence, B^+ only occurs when no training points fall in R^+. We see that, if R^+ has size

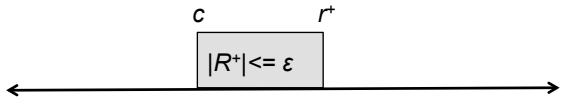

Figure 2: Bounding the region of an h too far to the right

ϵ, $\Pr[x_1 \notin R^+] \leq 1 - \epsilon$. Note: this is not an equality to account for the possibility of a probability mass at r^+. Given m training examples, we have

$$\begin{aligned}\Pr[B^+] &= \Pr[x_1 \notin R^+ \wedge ... \wedge x_m \notin R^+] \\ &= \Pr[x_1 \notin R^+]...\Pr[x_m \notin R^+] && \text{(by i.i.d assumptions)} \\ &\leq (1-\epsilon)^m\end{aligned}$$

Applying this logic to reason about both bad cases:

$$\begin{aligned}\Pr[\text{err}_D(h) > \epsilon] &\leq \Pr[B^+ \vee B^-] \\ &\leq \Pr[B^+] + \Pr[B^-] && \text{(by union bound)} \\ &\leq 2(1-\epsilon)^m && \text{(by symmetry)} \\ &\leq 2e^{-\epsilon m} && \text{(by } 1 + x \leq e^x\text{)}\end{aligned}$$

We can make the right side smaller than δ which, solving for m, will hold true if $m \geq \frac{1}{\epsilon}\ln\frac{2}{\delta}$. Hence, if we have at least m training examples according to this inequality, we can bound the generalization error with a given confidence level as desired. This approach of requesting some amount of data to provide some accuracy guarantees is nice theoretically, but not always realistic. In practice, we would likely receive some data and want to reason about the error on future data. Rearranging terms, we see what with probability $\geq 1 - \delta$, $\text{err}_D(h) \leq \frac{1}{m}\lg\frac{2}{\delta}$.

1.3 Other PAC learning examples

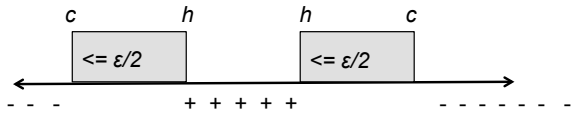

Figure 3: Learning intervals

As a second example, let $X = \mathbb{R}$ and $\mathcal{C} = \{\text{intervals}\}$, as illustrated in Figure 3. Specifically, a $c \in \mathcal{C}$ is a concept which is equal to 1 for x in some interval $[a, b]$ and 0 otherwise. Here, we have 2 boundary error regions, which we want force to have size of at most $\frac{\epsilon}{2}$ each. Overall, there are 4 possible bad events, which using similar analysis as before, requires $m \geq \frac{2}{\epsilon}\ln\frac{4}{\delta}$ examples to provide the desired accuracy with the desired level of confidence.

Finally, let $X = \mathbb{R}^2$ and $\mathcal{C} = \{\text{axis-aligned rectangles}\}$, as illustrated in Figure 4. Our hypothesis will be a rectangle that contains just the positive points. Extending the argument

used for the class of intervals, we have 4 possible regions of error, each of which we wish to bound to be of less than size $\frac{\epsilon}{4}$.

We can generalize this argument to $X = \mathbb{R}^n$ and $\mathcal{C} = \{\text{axis-aligned hyper-rectangles}\}$ to find a bound for m that is polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$ and n.

In general, we allow a PAC learning algorithm to use a number of examples (and also running time, if we are considering efficiency) that is polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, n, the "size" of the examples and the "size" of the concept c. For instance, when the examples are in $\mathbb{R}^n$ or $\{0,1\}^n$, we allow the time/sample to depend on n. And when learning, DNF formulas, we may allow time/sample size to depend on the number of terms in the target DNF.

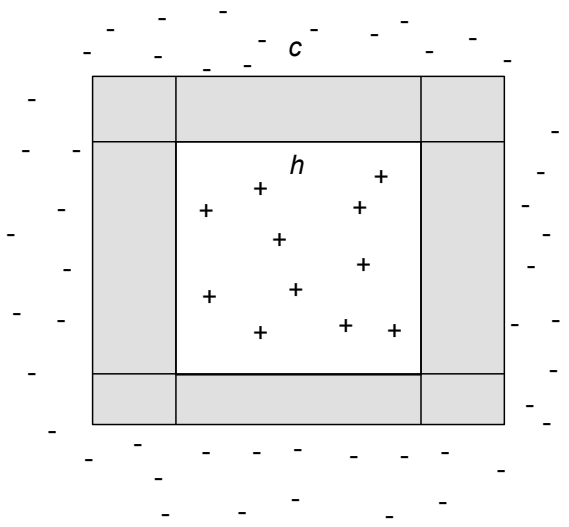

Figure 4: Learning squares

2 Return to Consistency

For now, we assume that $|\mathcal{H}| < \infty$. In the future, we will discuss cases in which the hypothesis space is not necessarily finite.

Theorem: Suppose that algorithm A finds a hypothesis $h_A \in \mathcal{H}$ which is consistent with all m training examples where $m \geq \frac{1}{\epsilon}(\lg|\mathcal{H}| + \lg\frac{1}{\delta})$. Then, $\Pr[\text{err}_D(h_A) > \epsilon] \leq \delta$ (i.e. it fulfills the PAC conditions).

Equivalently, with probability $\geq 1 - \delta$, if $h_A \in \mathcal{H}$ is consistent will all m examples, then

$$\text{err}_D(h_A) \leq \frac{\lg|\mathcal{H}| + \lg\frac{1}{\delta}}{m}$$

With respect to our previous error bound for PAC learning, we now add the size of the hypothesis space to our analysis. We consider $\lg|\mathcal{H}|$ to be a reasonable measure for hypothesis complexity since, if we were to give a bit-string identifier to each hypothesis, all such labels could be represented as a bit string of length $\lg|\mathcal{H}|$. As will be seen in the following experiment, the bigger the hypothesis space, the more likely it is that a very poor hypothesis appears spuriously on the training set to be good, purely by chance.

2.1 Labeling coin flips

The class was asked to provide labels $\in \{0, 1\}$ for the numbers 1-20. The first 10 were training examples and the next 10 were test examples. These results were compared to the outcome of 20 coin flips. The best guess in the class had a training error of 20% and a corresponding test error of 30%. In years past, this experiment has also led to hypotheses with training errors close to 0% and testing errors of 50%. These trials illustrate that, given a large hypothesis space, a consistent hypothesis may perform extremely well with the training examples only to perform poorly on the test input.

2.2 Additional Thoughts on Occam's Razor

When considering $\mathcal{H} = \{\text{monotone conjunctions}\}$, we see that $|\mathcal{H}| = 2^n$ so $\lg|\mathcal{H}| = n\lg 2$, which could very plausibly be smaller than m and therefore yield a useful bound on the error.

Next, consider $\mathcal{H} = \{\text{DNF}\}$. If terms are of size n and we build our hypothesis using m examples, the number of bits needed to build a hypothesis is $O(nm)$. This yields a bound for the error of $\frac{O(nm)+\lg\frac{1}{\delta}}{m} > 1$, which is true but entirely useless. This is in accordance to our intuition from the previous class that learning DNF by essentially memorizing the positive examples yields poor results due to the sheer size of the hypotheses (which are as large as the original dataset).

Overall, the principle of, all else being equal, attempting to minimize hypothesis complexity is known as Occam's Razor. Having discussed the ideal properties of simple consistent hypotheses, our next task is to discuss the computational problem of actually finding them.