COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Shaowei Zhu

Lecture #7
February 25, 2019

1 Sauer's Lemma

Sauer's lemma gives a way to bound the growth function using VC-dimension, enabling us to calculate how many training examples do we need in the PAC-learning setting when we are trying to learn an infinite hypothesis class.

Theorem 1 (Sauer's Lemma). *Denote the hypothesis space by $\mathcal{H}$. Let $d = \mathrm{VCdim}(\mathcal{H})$. Then the growth function for a set of m points satisfies*

$$\Pi_{\mathcal{H}}(m) \leq \sum_{i=0}^{d} \binom{m}{i} = \Phi_d(m) \tag{1}$$

Proof. Continue from where we left off in the previous lecture, we are proving this by induction on $m + d$. Let $S = \langle x_1, \ldots, x_m \rangle$ be any sequence of m examples. For illustration, we will consider the following set S consisting of $m = 5$ examples. The leftmost column contains all possible labelings of S given by some hypothesis class $\mathcal{H}$.

Let $S' = \langle x_1, \ldots, x_{m-1} \rangle$ be the sequence obtained from S by throwing away x_m. We define two new spaces of hypotheses $\mathcal{H}_1$ and $\mathcal{H}_2$. Formally, these spaces consist of hypotheses defined on the domain $\mathcal{X}' = \{x_1, \ldots, x_{m-1}\}$, but we will focus on their labelings on the set S'. If $\mathcal{H}$ induces two labelings of S with their first $(m-1)$ bits being the same in the leftmost column (like 01100 and 01101), then this string of $(m-1)$ bits goes into the set of possible labelings realized by both $\mathcal{H}_1$ and $\mathcal{H}_2$. Otherwise for those labelings in $\mathcal{H}$ that have the first $(m-1)$ bits being unique (appearing as part of a labeling only once, like 0111), those first $(m-1)$ bits go into only the set of possible labelings realized by $\mathcal{H}_1$ but not $\mathcal{H}_2$. See the example.

$\Pi_{\mathcal{H}}(S)$	$\mathcal{H}_1 = \Pi_{\mathcal{H}_1}(S')$	$\mathcal{H}_2 = \Pi_{\mathcal{H}_2}(S')$
01100 01101	0110	0110
01110	0111	
10010 10011	1001	1001
11001	1100	

We have proved the following two claims in the previous lecture:

Claim 1. $|\Pi_{\mathcal{H}}(S)| = |\mathcal{H}_1| + |\mathcal{H}_2|$.

Claim 2. $\mathrm{VCdim}(\mathcal{H}_1) \leq d$. Thus $|\mathcal{H}_1| = |\Pi_{\mathcal{H}_1}(S')| \leq \Phi_d(m-1)$ by inductive hypothesis. Here we can use the inductive hypothesis since we have reduced the size of the point set to $m-1$ while keeping VC-dimension the same.

Now we want to prove this third claim:

Claim 3. $\mathrm{VCdim}(\mathcal{H}_2) \leq d - 1$.

Suppose that $\mathrm{VCdim}(\mathcal{H}_2) = t$. Suppose that a set T containing t examples is shattered by $\mathcal{H}_2$. We now create a set $T' = T \cup \{x_m\}$, by adding the example x_m to T. We want to argue next that T' is shattered by $\mathcal{H}$.

By definition of shattering, $\mathcal{H}_2$ could realize all possible labelings for the t examples in T. For each labeling it realizes, we create two new labelings by appending a 0 and a 1 to it, respectively. Let this set of new labelings be L'. L' must include all possible labelings for the $(t+1)$ examples in T', since the last example could only be given 0 or 1. Thus we only need to show that $\mathcal{H}$ could realize all of L'.

From the definition of $\mathcal{H}_2$, we could see that for every labeling s realized by $\mathcal{H}_2$, $\mathcal{H}$ can realize two labelings $s0$ and $s1$. These two labelings agree with s on any example but x_m. Therefore, $\mathcal{H}$ can realize all of L' since L' is obtained from appending 0 or 1 to all labelings of T realized by $\mathcal{H}_2$. In other words, $\mathcal{H}$ could realize all possible labelings for this set T' of $(t+1)$ examples. Thus $\text{VCdim}(\mathcal{H}) \geq t+1 = \text{VCdim}(\mathcal{H}_2)+1$.

From Claim 3 we use our inductive hypothesis and get

$$|\mathcal{H}_2| = |\Pi_{\mathcal{H}_2}(S')| \leq \Phi_{d-1}(m-1) \tag{2}$$

Using the three claims we proved above and the recurrence property of binomial coefficients (introduced in the previous lecture), we get

$$\begin{aligned}
|\Pi_{\mathcal{H}}(S)| &= |\mathcal{H}_1| + |\mathcal{H}_2| && (3)\\
&\leq \sum_{i=0}^{d}\binom{m-1}{i} + \sum_{i=0}^{d-1}\binom{m-1}{i} && (4)\\
&= \sum_{i=0}^{d}\binom{m-1}{i} + \sum_{i=1}^{d}\binom{m-1}{i-1} && (5)\\
&= \sum_{i=0}^{d}\binom{m-1}{i} + \sum_{i=0}^{d}\binom{m-1}{i-1} && (6)\\
&= \sum_{i=0}^{d}\left[\binom{m-1}{i} + \binom{m-1}{i-1}\right] && (7)\\
&= \sum_{i=0}^{d}\binom{m}{i} && (8)\\
&= \Phi_d(m) && (9)
\end{aligned}$$

□

The following fact is a more exact bound for $\Phi_d(m)$ other than $O(m^d)$. Its proof could be found in the textbook.

Lemma 1 (An estimation for $\Phi_d(m)$)**.** *If $m \geq d \geq 1$ then*

$$\Phi_d(m) \leq \left(\frac{em}{d}\right)^d \tag{10}$$

2 Lower bound on number of examples for PAC-learnability

In the last lecture we have shown that having roughly $O(d)$ examples is a sufficient condition for PAC-learning where d is the VC-dimension of the hypothesis class. Here we want to argue that having $\Omega(d)$ examples is also a necessary condition for PAC-learnability (or PAC learning is not possible when you have less than $O(d)$ examples).

Intuitively, if the target concept can be very complex, then having very few examples could reveal very little about the nature of the target concept. In this case, knowing how the concept labels a portion of the domain could be of little help when we want to predict how the concept labels other portions of the domain, since the concept is complex enough to potentially label other portions in an arbitrarily complex way. Thus whatever learning algorithm you use or whatever hypothesis you use, there is a great chance for it to generalize poorly outside of the training examples, due to the complex nature of the underlying concept.

The above arguments hint that we should focus on the complexity of the concept class when trying to deduce a lower bound. We could slightly formalize the thoughts above as a concrete example. Let $d = \text{VCdim}(\mathcal{C})$, where $\mathcal{C}$ is the concept class that contains the target concept. Then there must exist a set $\{z_1, z_2, \ldots, z_d\}$ that is shattered by $\mathcal{C}$. If we only get to see how $c \in \mathcal{C}$ labels some of these points, it is impossible for us to infer how c labels other points, since all possible labelings could be realized by $\mathcal{C}$. We will try to formalize this intuition for the rest of this section.

The following statement and "proof" is a first (failed) attempt that tries to establish a lower bound for PAC-learnability.

Statement 1 (Lower bound of examples necessary for PAC-learnability, first attempt)**.** *For any learning algorithm A that learns the concept class $\mathcal{C}$, if given too few examples $m \le d/2$, then* $\text{err}(h_A) > 1/8$.

Proof. Attention: this proof is incorrect! Let D be the uniform distribution on $z_1, \ldots, z_d$. Run algorithm A with sample set S consisting of $m \le d/2$ examples that have been labeled arbitrarily and obtain h_A.

Let $c \in \mathcal{C}$ be a concept that is consistent with S and their labels, but $c(x) \neq h_A(x)$ for any $x \notin S$. Obviously, h_A will be incorrect on every x outside of S, and thus $\text{err}(h_A) \ge 1/2$ since the training set S covers no more than half of the distribution. □

The proof above is completely wrong, because in the PAC learning setting the concept c has to be fixed before we select the training set and their labels. We have to be more careful when formulating the theorem and the proof.

Theorem 2 (Lower bound of examples necessary for PAC-learnability)**.** *Suppose that* $\text{VCdim}(\mathcal{C}) = d$. *For any learning algorithm A, there exists a target concept $c \in \mathcal{C}$ and distribution D such that if A only gets random sample set S of size $m \le d/2$, then*

$$\Pr_S\left[\text{err}_D(h_A) > \frac{1}{8}\right] \ge \frac{1}{8} \tag{11}$$

We could compare this with the usual PAC learning criterion, which says if we have a sufficient amount of examples m then we could bound the probability for obtaining a bad h_A. This theorem says that if m is not large enough, then the probability for you to obtain a bad h_A will be larger than some constant. More intuitively, for any PAC-learning algorithm for $\mathcal{C}$, if $\epsilon \le 1/8$ and $\delta \le 1/8$, then the algorithm requires at least $d/2$ training examples.

Proof. Let $\mathcal{C}'$ be a set of concepts that contains one representative for each labeling of the shattered set $\{z_1, \ldots, z_d\}$. By doing this we obtain a finite concept class which satisfies $|\mathcal{C}'| = 2^d$ that is easier to work with.

Let D be uniform distribution on $z_1, \ldots, z_d$. Choose c uniformly at random from $\mathcal{C}'$. Draw S according to D and label every example in S according to c.

Suppose that A is a deterministic algorithm that gives us h_A, after seeing S. Now consider the probability for h_A to be incorrect on a random point x drawn according to D (the variables under the Pr symbol indicate what is the probability over; note that there is randomness in selecting c, S, x):

$$\Pr_{c,S,x}[h_A(x) \neq c(x)] \tag{12}$$
$$\geq \Pr_{c,S,x}[h_A(x) \neq c(x) \wedge x \notin S] \tag{13}$$
$$= \Pr_{c,S,x}[x \notin S] \cdot \Pr_{c,S,x}[h_A(x) \neq c(x) \mid x \notin S] \tag{14}$$
$$\geq \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4} \tag{15}$$

Here we arrive at Equation 13 by the implication law in probability (if $A \Rightarrow B$ then $\Pr(A) \leq \Pr(B)$). We obtain Equation 15 since $\Pr_{c,S,x}[x \notin S] \leq 1/2$ (when selecting x there are d examples to choose from and S has less than $d/2$ examples) and $\Pr_{c,S,x}[h_A(x) \neq c(x) \mid x \notin S] = 1/2$. This latter equation deserves a little more explanation. $h_A(x)$ does depend on x, the samples in S, and their labels, but it does not depend on $c(x)$ when conditioned on $x \notin S$ because the only x and $c(x)$ seen by A are those in S. Therefore, the random variable $c(x)$ is independent of $h_A(x)$ when conditioned on $x \notin S$. And since the label $c(x)$ is chosen uniformly at random from $\{0, 1\}$, the chance that it agrees with $h_A(x)$ is exactly $1/2$.

We have obtained that

$$\mathbb{E}_c\left[\Pr_{S,x}[h_A(x) \neq c(x)]\right] = \Pr_{c,S,x}[h_A(x) \neq c(x)] \geq \frac{1}{4}. \tag{16}$$

Thus there exists $c \in \mathcal{C}$ such that

$$\frac{1}{4} \leq \Pr_{S,x}[h_A(x) \neq c(x)] \tag{17}$$
$$= \mathbb{E}_S\left[\Pr_x[h_A(x) \neq c(x)]\right] \tag{18}$$
$$= \mathbb{E}_S[\mathrm{err}_D(h_A)] \tag{19}$$
$$= \Pr_S\left[\mathrm{err}_D(h_A) > \frac{1}{8}\right] \mathbb{E}_S\left[\mathrm{err}_D(h_A) \mid \mathrm{err}_D(h_A) > \frac{1}{8}\right] + \Pr_S\left[\mathrm{err}_D(h_A) \leq \frac{1}{8}\right] \mathbb{E}_S\left[\mathrm{err}_D(h_A) \mid \mathrm{err}_D(h_A) \leq \frac{1}{8}\right] \tag{20}$$
$$\leq \Pr_S\left[\mathrm{err}_D(h_A) > \frac{1}{8}\right] \cdot 1 + 1 \cdot \mathbb{E}_S\left[\mathrm{err}_D(h_A) \mid \mathrm{err}_D(h_A) \leq \frac{1}{8}\right] \tag{21}$$
$$\leq \Pr_S\left[\mathrm{err}_D(h_A) > \frac{1}{8}\right] + \frac{1}{8} \tag{22}$$

Hence $\Pr_S\left[\mathrm{err}_D(h_A) > \frac{1}{8}\right] \geq \frac{1}{4} - \frac{1}{8} = \frac{1}{8}$. □

3 What if there is no consistent hypothesis?

So far we have been focusing on learning when we can find a hypothesis that is consistent with the training examples. What if we cannot find one? Several things could be preventing us from finding the consistent hypothesis here:

1. The hypothesis class $\mathcal{H}$ might not capture all of the concept class $\mathcal{C}$. Thus for some $c \in \mathcal{C}$ there is no satisfactory hypothesis.

2. There is no computationally efficient algorithm that can find a consistent hypothesis.

3. Maybe there is no functional relationship between instances and their labels (e.g., there is noise or other random factors involved when labels are generated).

In order to discuss learning under the non-consistent setting, we need to generalize our notions a little bit: suppose that every training and test example (x, y) is drawn from a joint distribution $\mathcal{X} \times \{0, 1\}$. We could also view the process that generates data as a two-step process: it first chooses x according to some distribution, and then it assigns a label y to x according to some other distribution conditioned on x, as reflected by the formula $\Pr[(x, y)] = \Pr[x] \cdot \Pr[y \mid x]$.