COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Karen Feng

Lecture #4
February 14, 2018

Review

Let h denote a hypothesis from the space $\mathcal{H}$ of size $|\mathcal{H}|$. Let h_A denote a hypothesis found by the algorithm A. Let c denote a concept from the concept class $\mathcal{C}$. Let m denote the size of a training set $\mathcal{S}$. Let x_i denote the i-th sample from the training set, and $c(x_i)$ denote the label corresponding to the sample. We assume that the examples are drawn from a target distribution $x_i \sim D$.

The generalization error of a hypothesis over the target distribution is:

$$\text{err}_D(h) = \Pr[h(x) \neq c(x)]$$

A hypothesis h is consistent if it correctly labels all examples in a training set according to the target concept c: if $h(x_i) = c(x_i)\ \forall i \in \{1, \ldots, m\}$.

1 Occam's Razor

Occam's Razor provides a general technique to show that a learning algorithm generalizes with low error to new data with high probability — is probably approximately correct (PAC).

Theorem (Occam's Razor)**.** *Say algorithm A 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:*

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

In other words, the probability that the generalization error of h_A is ϵ-bad is bounded by δ.

Theorem (Equivalent)**.** *With probability at least $1 - \delta$, if $h_A \in \mathcal{H}$ is consistent, then its true generalization error is bounded:*

$$err_D(h_A) \leq \frac{\ln|\mathcal{H}| + \ln\frac{1}{\delta}}{m}$$

The generalization error decreases inversely to the number of training examples m. It also decreases if we know more about the target concept c being learned. If we know that c belongs to a smaller class $\mathcal{C}$, we often can choose a correspondingly smaller hypothesis space $\mathcal{H}$ so that the $\ln|\mathcal{H}|$ term will also be smaller. In addition, we previously saw that $\ln|\mathcal{H}|$ is proportional to the number of bits needed to describe hypotheses in $\mathcal{H}$. This implies:

Learning is possible if we can find a consistent hypothesis that is simple, and if given enough data.

2 Learnability in the Consistency Model Implies PAC-Learnability

Proposition. *If we can find a consistent hypothesis, then PAC learning is possible.*

Let us suppose that our algorithm finds a hypothesis that is ϵ-bad with error rate 1% ($\epsilon = 0.01$). If there are hundreds of training examples, our hypothesis will almost certainly make at least one mistake and thus not be consistent. We make the following remark by generalizing this to all ϵ-bad hypotheses in the hypothesis space.

Remark. *With high probability, any ϵ-bad hypotheses will be eliminated from our training set, and any remaining hypotheses will be ϵ-good.*

Proof. We demonstrate that the probability of a consistent hypothesis being ϵ-bad is bounded by δ.

Let us denote the set of ϵ-bad hypotheses $\mathcal{B} = \{h \in \mathcal{H} : h \text{ } \epsilon\text{-bad}\}$. We calculate the probability that some $h \in \mathcal{B}$ is consistent on all m examples.

$$\begin{aligned}
\Pr[h \text{ consistent}] &= \Pr[h(x_1) = c(x_1) \wedge \cdots \wedge h(x_m) = c(x_m)] \\
&= \prod_{i=1}^{m} \Pr_{x_i \in D}[h(x_i) = c(x_i)] && (2.1) \\
&\leq (1-\epsilon)^m && (2.2) \\
&\leq e^{-\epsilon m} && (2.3)
\end{aligned}$$

Equation 2.1 depends on the assumption that the training examples x_i are drawn independently from D. Based on our assumption that h is ϵ-bad, equation 2.2 uses the fact that $\Pr[h(x_i) = c(x_i)] \leq 1 - \epsilon$. Equation 2.3 uses the general bound that $1 + x \leq e^x \ \forall x$.

We calculate the probability that a consistent hypothesis h_A is ϵ-bad.

$$\begin{aligned}
\Pr[h_A \text{ consistent \& } \epsilon\text{-bad}] &\leq \Pr[\exists h \in \mathcal{H} : h \text{ consistent \& } \epsilon\text{-bad}] && (2.4) \\
&= \Pr[\exists h \in \mathcal{B} : h \text{ consistent}] \\
&= \Pr[\bigvee_{h \in \mathcal{B}} h \text{ consistent}] \\
&\leq \sum_{h \in \mathcal{B}} \Pr[h \text{ consistent}] && (2.5) \\
&\leq |\mathcal{B}| e^{-\epsilon m} && (2.6) \\
&\leq |\mathcal{H}| e^{-\epsilon m} && (2.7) \\
&\leq \delta && (2.8)
\end{aligned}$$

Equation 2.4 uses the general fact that if $a \Rightarrow b$, then $\Pr[a] \leq \Pr[b]$. Equation 2.5 is based on the union bound. We get equation 2.6 using the probability bound for ϵ-bad hypotheses being consistent found in equation 2.3. Equation 2.7 uses the fact that the set of bad hypotheses $\mathcal{B}$ is contained in the hypothesis space $\mathcal{H}$. Equation 2.8 assumes that the training set size $m \geq \frac{1}{\epsilon}(\ln |\mathcal{H}| + \ln \frac{1}{\delta})$.

□

We note that as long as the complexity of the hypothesis space $\ln |\mathcal{H}|$ does not grow faster than the size of the training set, we have a useful bound.

3 A "Better" Bound

The bound in Occam's Razor depends on the size of the entire hypothesis space $|\mathcal{H}|$. We consider whether it is possible to improve our bounds by considering only one consistent hypothesis h_A. The following attempt at a proof turns out to have *several errors*.

$$\begin{aligned}
\Pr[h_A \text{ } \epsilon\text{-bad} | h_A \text{ consistent}] &= \frac{\Pr[h_A \text{ } \epsilon\text{-bad \& consistent}]}{\Pr[h_A \text{ consistent}]} \\
&= \Pr[h_A \text{ } \epsilon\text{-bad \& consistent}] && (3.1) \\
&= \Pr[h_A \text{ consistent} | h_A \text{ } \epsilon\text{-bad}] \cdot \Pr[h_A \text{ } \epsilon\text{-bad}] && (3.2) \\
&\leq \Pr[h_A \text{ consistent} | h_A \text{ } \epsilon\text{-bad}] && (3.3) \\
&= \Pr[h_A(x_1) = c(x_1) \wedge \cdots \wedge h_A(x_m) = c(x_m) | h_A \text{ } \epsilon\text{-bad}] \\
&= \prod_{i=1}^{m} \Pr[h_A(x_i) = c(x_i) | h_A \text{ } \epsilon\text{-bad}] && (3.4) \\
&\leq (1-\epsilon)^m && (3.5) \\
&\leq e^{-\epsilon m} && (3.6) \\
&\leq \delta && (3.7)
\end{aligned}$$

Equation 3.1 uses the assumption that as h_A is consistent, $\Pr[h_A \text{ consistent}] = 1$. Equation 3.2 comes from the definition of conditional probability. As no probabilities are bigger than one, 3.3 comes from $\Pr[h_A \text{ } \epsilon\text{-bad}] \leq 1$. Equation 3.4 uses the assumption that the events are independent. We reach equation 3.5 because we are conditioning on h_A being ϵ-bad. Equation 3.6 uses the general bound that $1 + x \leq e^x \ \forall x$. Equation 3.7 assumes that the training set size $m \geq \frac{1}{\epsilon}(\ln \frac{1}{\delta})$.

We seem to get a better bound on the number of training examples m without depending on $|\mathcal{H}|$ — but it's too good to be true! Where did we go wrong?

We falsely assumed in equation 3.4 that the training examples are independent. The events $h(x_1) = c(x_1)$ and $h(x_2) = c(x_2)$ are independent events if h is a fixed hypothesis. As h_A is a random variable dependent on the training set $\mathcal{S}$, the events are no longer independent — instead, they depend on the entire training set. In addition, equation 3.5 is incorrect as our initial assumption that h_A is consistent dictates that $\Pr[h_A(x_i) = c(x_i) | h_A \text{ } \epsilon\text{-bad}] = 1$ rather than $\leq 1 - \epsilon$. Again, the problem is that h_A is a random variable, not a fixed hypothesis.

In short, we need to decide which hypotheses are being considered *before* choosing the random training sample.

4 PAC-Learnability Implies Learnability in the Consistency Model

We demonstrated in section 2 that learnability in the consistency model implies PAC-learnability in the case that the hypothesis space is finite. We now consider the converse.

Proposition. *If we can do proper PAC learning, then we can do learning in the consistency model.*

Proof. Suppose we are given an algorithm A that properly PAC-learns $\mathcal{C}$ (by properly, we mean that $\mathcal{H} = \mathcal{C}$). In other words, the algorithm takes as input random examples from some distribution D and outputs a hypothesis $h \in \mathcal{C}$ such that $\text{err}_D(h) \leq \epsilon$ with probability $1 - \delta$. We are also given a set of labeled examples $\mathcal{S} = \langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$. These examples are *not* random and are just given to us.

Our goal is to use the algorithm A to somehow solve the consistency problem, i.e., find a concept in $\mathcal{C}$ consistent with $\mathcal{S}$ or say that no such concept exists.

We construct D as a uniform distribution over $\mathcal{S}$, and choose $\epsilon = \frac{1}{2m} < \frac{1}{m}$. We choose as many random examples from D as are needed by A. The algorithm A then outputs a hypothesis h. If h is consistent with $\mathcal{S}$, then we return it, since that's what we were looking for. Otherwise, we say that there is no consistent concept in $\mathcal{C}$.

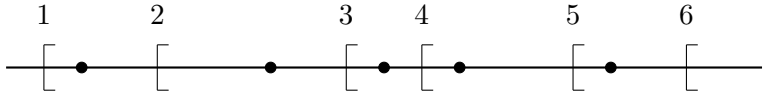
To argue correctness, we consider the case that $\exists c \in \mathcal{C}$ consistent with $\mathcal{S}$. Based on our assumption that A is a PAC-learning algorithm, with probability $\geq 1 - \delta$ we will find $h \in \mathcal{C}$ such that $\text{err}_D(h) \leq \epsilon < \frac{1}{m}$. If h makes any mistakes on $\mathcal{S}$, the probability of $\text{err}_D(h_A) \geq \frac{1}{m}$ as D is uniform over $\mathcal{S}$; this is a contradiction. Therefore we find that h is consistent with $\mathcal{S}$.

It is trivially true that in the case that $\nexists c \in \mathcal{C}$, we are always correct. □

5 PAC Learning with Infinitely Large Hypothesis Space

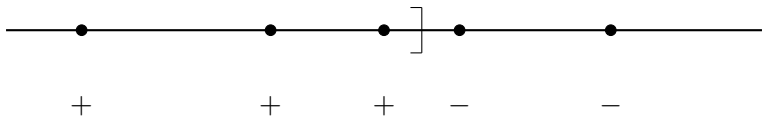
We have seen that Occam's Razor can be used if we have a finitely large hypothesis space, such as that of monotone conjunctions. We cannot apply Occam's Razor if we have an infinitely large hypothesis space, such as intervals on a line, axis-aligned rectangles, and positive half-lines. However, we have seen that we can do PAC learning on such problems. What makes it possible?

Example (Positive Half-Lines)**.** *Consider positive half-lines over a set of $m = 5$ data points. Although there are infinitely many hypotheses, there are only $m + 1 = 6$ different labelings (also known as behaviors or dichotomies).*

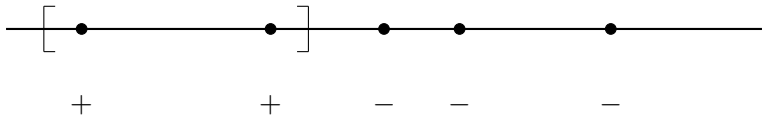


1	+	+	+	+	+
2	−	+	+	+	+
3	−	−	+	+	+
4	−	−	−	+	+
5	−	−	−	−	+
6	−	−	−	−	−

Example (Lines)**.** *Consider half-lines (either positive or negative) over a set of m data points. There are only $2(m+1) - 2 = 2m$ labelings. We calculate this by doubling the $m+1$ labelings from the previous example (flipping $+/-$ for the positive/negative half-lines) and subtracting the 2 overcounted all $+/-$ examples.*



Example (Intervals)**.** *Consider intervals over a set of m data points. There are only $\binom{m}{2} + m + 1$ labelings. We calculate this by choosing $\binom{m}{2}$ two distinct points as end-points, choosing m singletons, or letting all points be $-$.*



Over a finite number of data points, there are a finite number of labelings. In the general case of unlabeled points $\mathcal{S} = \langle x_1, \ldots, x_m \rangle$, the set of labelings is:

$$\Pi_{\mathcal{H}}(\mathcal{S}) = \{\langle h(x_1), \ldots, h(x_m) : h \in \mathcal{H} \rangle\}$$

Definition (Growth function)**.** *The size of the set of labels is the growth function:*

$$\Pi_{\mathcal{H}}(m) = \max_{\mathcal{S}:|\mathcal{S}|=m} |\Pi_{\mathcal{H}}(\mathcal{S})|$$