COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Amlan Sinha

Lecture #4
February 13, 2019

1 Occam's Razor

Occam's Razor is the principle which states that all else being equal, we expect simpler hypothesis to perform better than more complicated ones. As shown later in this lecture, this theorem bounds the generalization error for a learning algorithm — the bound is a function of the measure of the complexity of the hypotheses being used, namely $\ln|\mathcal{H}|$. This is why, the name *Occam's Razor* has sometimes been tied to this theorem.

In this document, h denotes a hypothesis drawn from hypothesis space $\mathcal{H}$, h_A denotes a hypothesis found by algorithm A, c denotes a concept drawn from concept class $\mathcal{C}$, m denotes the size of the training set S, ϵ denotes the error and δ denotes the confidence. We assume that both the training and the testing examples are drawn from an independent and identically distributed (*i.i.d.*) target distribution D, and we let x_i denote the i^{th} sample from S and $c(x_i)$ denote the label corresponding to the sample.

Example: c = Prof. Schapire, h = students

Students were asked to write a bit string with a length of 20. The first ten in the set were considered to be training samples while the next ten were considered to be the test set. Then, Prof. Schapire wrote a bit string on the board, which was also split into the training set and the test set in a similar fashion. Students were asked to compare their sequence in the training set with that of Prof. Schapire's and their sequence in the test set with that of Prof. Schapire's. The number of instances where the sequence did not match was used to estimate the error percentage. Even in a class size of 95 people, it was possible to achieve an error percentage as low as 20% on the training set. However, this hypothesis had a much worse error of 40% on the test set. In fact, all of the labels had been chosen at random by flipping a coin. Because the class is fairly large, it was very likely that at least one student would do very well, by sheer luck, as actually happened. However, when that one student was asked to predict the test labels, which were random, the expected number of mistakes was 50%. The point is that when working with a large hypothesis space, it becomes very likely that a poor predictor will appear to be good by luck, and as a result, the larger the hypothesis space, the worse the error bound.

1.1 Theorem 1.

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

1.2 Theorem 2 (equivalent to Theorem 1).

Suppose algorithm A finds a hypothesis $h_A \in \mathcal{H}$ consistent with m examples. Then it can be said with probability at least $1 - \delta$ that $err_D(h_A) \leq \frac{\ln\mathcal{H} + \ln\frac{1}{\delta}}{m}$.

It is important to note that the generalization error is inversely proportional to the training sample size — the more training samples we use, the lower the generalization error will be. If we have a large hypothesis space, we require more training samples to remove the bad hypotheses with high probability. However, the smaller the size $|\mathcal{H}|$ of the hypothesis space is (or the more we know about the concept class $\mathcal{C}$), the fewer training samples we require.

2 Learnability in the Consistency model implies learnability in the PAC model

Proof

Let

$$\epsilon = \frac{\ln \mathcal{H} + \ln \frac{1}{\delta}}{m},$$

and let $\mathcal{B} = \{h \in \mathcal{H} : h \textit{ is } \epsilon\textit{-bad}\}$ denote the hypotheses $h \in \mathcal{H}$ which are *ϵ-bad* meaning $err_D(h) > \epsilon$. Assuming h_A to be a consistent hypothesis found by algorithm A, we attempt to show that $Pr[err_D(h_A) > \epsilon] \leq \delta$. In other words, we attempt to show that, with high probability, if h_A is consistent, then it is *ϵ-good.* That is, we want to show

$$Pr[h_A \textit{ consistent } \Rightarrow h_A \ \epsilon\textit{-good}] \geq 1 - \delta$$

which is equivalent to showing

$$Pr[h_A \textit{ consistent } \wedge h_A \ \epsilon\textit{-bad}] \leq \delta.$$

This is because if A and B are events, then $A \Rightarrow B$ is exactly equivalent to $(\neg A) \vee B$ whose negation is $A \wedge (\neg B)$. We can bound this as follows:

$$\begin{aligned}
Pr[h_A \textit{ consistent } \wedge h_A \epsilon\textit{-bad}] &\leq Pr[\exists h \in \mathcal{H} :\ h \textit{ consistent } \wedge h \epsilon\textit{-bad}] && (1)\\
&= Pr[\exists h \in \mathcal{B} :\ h \textit{ consistent}] && (2)\\
&= Pr\left[\bigvee_{h \in \mathcal{B}} h \textit{ consistent}\right] && (3)\\
&\leq \sum_{h \in \mathcal{B}} Pr[h \textit{ consistent}] && (4)\\
&= \sum_{h \in \mathcal{B}} Pr[h(x_1) = c(x_1) \wedge \cdots \wedge h(x_m) = c(x_m)] && (5)\\
&= \sum_{h \in \mathcal{B}} \prod_{i=1}^{m} Pr[h(x_i) = c(x_i)] && (6)\\
&\leq \sum_{h \in \mathcal{B}} (1 - \epsilon)^m && (7)\\
&= |\mathcal{B}|(1 - \epsilon)^m && (8)\\
&\leq |\mathcal{H}|(1 - \epsilon)^m && (9)\\
&\leq |\mathcal{H}| e^{-\epsilon m} && (10)\\
&= \delta && (11)
\end{aligned}$$

If A and B are events and $A \Rightarrow B$, then $Pr[A] \leq Pr[B]$. This yields Equation(1). By employing the definition of $\mathcal{B}$, we can easily get to Equation(2). Equation (4) is given by using the union bound, which states that $Pr[A \vee B] \leq Pr[A]+Pr[B]$. We use the definition of consistency which states that h is consistent if $[h(x_1) = c(x_1) \wedge \cdots \wedge h(x_m) = c(x_m)]$ to get to Equation(5). Because the examples are chosen i.i.d. from target distribution D, we can easily get to Equation (6) from Equation (5). Based on our assumption that h is ϵ*-bad*, $Pr[h(x_i) = c(x_i)] \leq (1 - \epsilon)$. This gives us Equation(7). Because $\mathcal{B} \subseteq \mathcal{H}$, we can easily get Equation (9) from Equation (8). Then, we use the general fact that $1 + x \leq e^x \ \forall \ x$ to get Equation (10) from Equation (9). The last line uses our choice of ϵ (which was chosen so that this equality would hold).

Remark:

In other words, this shows that with high probability, any ϵ-bad hypothesis will be eliminated from our training samples and the remaining hypotheses will be ϵ-good.

2.1 The 'wrong' proof

The bound on the generalization error as shown in the previous section depends on the size of the hypothesis space $|\mathcal{H}|$. We attempt to improve the bounds by considering only one consistent hypothesis h_A without using $|\mathcal{H}|$. As in the previous section, we want to bound $Pr[h_A \textit{ consistent } \wedge \ h_A \ \epsilon\textit{-bad}]$

$$\begin{aligned}
Pr[h_A \textit{ consistent } \wedge \ h_A \ \epsilon\textit{-bad}] &= Pr[h_A \textit{ consistent} \mid h_A \ \epsilon\textit{-bad}] \ Pr[h_A \ \epsilon\textit{-bad}] && (12)\\
&\leq Pr[h_A \textit{ consistent} \mid h_A \ \epsilon\textit{-bad}] && (13)\\
&= Pr[h_A(x_1) = c(x_1) \wedge \cdots \wedge h_A(x_m) = c(x_m) \mid h_A \ \epsilon\textit{-bad}] && (14)\\
&= \prod_{i=1}^{m} Pr[h_A(x_i) = c(x_i) \mid h_A \ \epsilon\textit{-bad}] && (15)\\
&\leq (1 - \epsilon)^m && (16)\\
&\leq e^{-\epsilon m} && (17)\\
&= \delta && (18)
\end{aligned}$$

where the last equality holds if we let $\epsilon = \ln(1/\delta)/m$. This means with probability at least $1 - \delta$, if the algorithm can find a consistent hypothesis h_A then that hypothesis is ϵ*-good.* The results seem too good to be true because it does not depend on the hypothesis at all! In comparison, the bound in the Occam's Razor theorem depended on the complexity of the hypothesis space.

The maximum probability of an event is always less than 1, and therefore, $Pr[h_A \ \epsilon\textit{-bad}] \leq 1$, giving us Equation (13). Then, we use the definition of consistency to get Equation (14). Then, we use the fact that the samples were drawn i.i.d. from the target distribution D to get Equation (15). Given that h_A is ϵ*-bad*, it follows that $\forall x_i \ Pr[h_A(x_i) = c(x_i)] \leq 1 - \epsilon$ giving us Equation (16).

Although the arguments seems believable, the proof is actually incorrect. The issue with the approach is that h_A is chosen as a function of S. It is therefore a random variable that depends on S. Since h_A is chosen using S, the samples are no longer i.i.d. and therefore

Equation (15) is no longer true. In addition, Equation (16) is also incorrect because h_A is chosen such that it is consistent which means that $Pr[h_A(x_i) = c(x_i) \mid h_A \text{ } \epsilon\text{-}bad] = 1$

3 Learnability in the (proper) PAC model implies learnability in the Consistency model

Previously we have shown that if a problem is learnable via the consistency model, we have a *proper* PAC learning algorithm for $\mathcal{C}$. Here, *proper* PAC learning refers to the case when $\mathcal{C} = \mathcal{H}$. Now, we are going to explore whether the converse is true. That is, *if there exists a proper PAC algorithm A for $\mathcal{C}$, given an arbitrary (**not random**) set of labeled examples $S = \langle (x_1, y_1), \cdots, (x_m, y_m) \rangle$ can we use A to find a target concept $c \in \mathcal{C}$ such that it is consistent with all the labeled examples given by S?* We aim to show that A solves the consistency problem by finding a concept $c \in \mathcal{C}$ consistent with all the examples in S or says that no such concept exists.

3.1 Proof

Suppose there exists an algorithm A that *properly* PAC-learns. The algorithm takes random examples from some distribution D as input and returns a hypothesis $h \in \mathcal{C}$ as output such that $Pr[err_D(h) > \epsilon] \leq \delta$.

We also are given a sample S and seek to use A to find a concept that is consistent with S. It is important to note that this set S is *not* random.

First, we construct a distribution D which is a uniform distribution over S. Then, we define $\epsilon = \frac{1}{2m} < \frac{1}{m}$ and $\delta > 0$. By running the algorithm A on m' random samples from D, where $m' = poly(\frac{1}{\epsilon}, \frac{1}{\delta})$ is the number of examples required by A, we obtain the hypothesis which we are going to denote as h. If h is consistent, we output h since it satisfies our goal of finding a consistent concept in $\mathcal{C}$. Otherwise, we say no such concept exists. To see that this construction works, if no consistent concept exists, then our algorithm will not find one, so it will correctly report that no consistent concept exists. Otherwise, let us consider the case when $\exists\, c \in \mathcal{C}$ consistent with S. Because A is a PAC-learning algorithm, it will find an $h \in \mathcal{C}$ since $\mathcal{H} = \mathcal{C}$ such that $Pr[err_D(h) > \epsilon] \leq \delta$ or in other words, $err_D(h) \leq \epsilon < \frac{1}{m}$ with probability $\geq 1 - \delta$. If h misclassifies a sample in S, then $err_D(h) \geq \frac{1}{m}$ since D is uniform over S, violating the assumption that h is ϵ-*good*. So, h has to be consistent with S (with probability at least $1 - \delta$).