0.1 Titolo

(Lesson 5 of 16/10/19) Compiled: October 16, 2019

A bit of clarification on the previous theorem.
Recall the last point of the previous demonstration:

$$L_D(h_S) \underset{(a)}{\leq} L_S(h_S) + \frac{\epsilon}{2} \underset{(b)}{\leq} L_S(h) + \frac{\epsilon}{2} \underset{(c)}{\leq} \left(L_D(h) + \frac{\epsilon}{2}\right) + \frac{\epsilon}{2}$$

For the various inequalities, one needs to reason in terms of "worst-cases". For the first inequality (a), if $L_D < L_S$ it works, otherwise, if $L_D > L_S$, then one applies the definition of ϵ-representative.
Then (b) comes from the ERM algorithm, and for (c) we have again two possibilities: if $_D < L_S$ we apply the definition of ϵ-representative, otherwise, if $L_D > L_s$, the inequality is already justified.

0.2 Finite classes are Agnostic PAC Learnable

Proof (part 2).
In the last lecture, we arrived at:

$$P_{\text{bad}} = D^m\left(\bigcup_{h\in\mathcal{H}} \{S\colon |L_S(h) - L_D(h)| > \epsilon\}\right) \leq \sum_{h\in\mathcal{H}} D^m(\{S\colon |L_S(h) - L_D(h)| > \epsilon\}) \tag{1}$$

Recall that:

$$L_S(h) = \frac{1}{m}\sum_{i=1}^{m} l(h, z_i) \qquad z_i \in S$$
$$L_D(h) = \mathbb{E}_{z\sim D}[l(h, z)]$$

Intuitively, by applying the *law of large numbers* the average in $L_S(h)$ will approach the expected value in $L_D(h)$.
More formally, we make us of the **Hoeffding's Inequality**.
Let $\theta_1, \ldots, \theta_m$ be a sequence of i.i.d. random variables and assume that for all i, $\mathbb{E}[\theta_i] = \mu$ and $\mathbb{P}[a \leq \theta_i \leq b] = 1$. Then, for any $\epsilon > 0$:

$$\mathbb{P}\left[\left|\frac{1}{m}\sum_{i=1}^{m}\theta_i - \mu\right| > \epsilon\right] \leq 2\exp\left(-\frac{2m\epsilon^2}{(b-a)^2}\right) \tag{2}$$

That is, the probability that the average and expected value are significantly different ($|\bar{\theta} - \mu| > \epsilon$) vanishes exponentially as the number m of samples increases.

In our case, $L_S(h)$ covers the role of $\bar{\theta}$, and $L_D(h)$ that of μ. Thus:

$$D^m(\{S\colon |L_S(h) - L_D(h)| > \epsilon\}) = \mathbb{P}\left[\left|\frac{1}{m}\sum_{i=1}^{m}\theta_i - \mu\right| > \epsilon\right] \underset{(2)}{\leq} 2e^{-2m\epsilon^2}$$

where we assumed as interval $[0,1]$ (meaning that the loss is bounded between $[0,1]$, as happens in binary classification, so that $(b-a)^2 = 1$).
Substituting this result back in (1) we get:

$$P_{\text{bad}} \leq \sum_{h \in \mathcal{H}} 2e^{-2m\epsilon^2} = |\mathcal{H}| 2e^{-2m\epsilon^2}$$

By setting $P_{\text{bad}} \overset{!}{\leq} \delta$ and rearranging:

$$|\mathcal{H}| 2e^{-2m\epsilon^2} \leq \delta \Rightarrow -2m\epsilon^2 \leq \ln\left(\frac{\delta}{2|\mathcal{H}|}\right) \Rightarrow m \geq \frac{1}{-2\epsilon^2} \ln\left(\frac{\delta}{2|\mathcal{H}|}\right) = \frac{1}{2\epsilon^2} \ln\left(\frac{2|\mathcal{H}|}{\delta}\right)$$

This theorem gives a *sufficient condition* for uniform convergence (and thus for agnostic PAC learnability).

0.2.1 Some comments

In many real world applications we consider hypothesis classes determined by a set of parameters in $\mathbb{R}$ - forming an *infinite* set of hypotheses.
However, in reality a computer represents numbers with a *finite* precision - and so the parameters are not real numbers, but result from a *fine discretization*. For example, if we use 64-bit precision, and d parameters, the hypothesis class has a cardinality of $|\mathcal{H}| = 2^{64d}$ - which is very very large, but finite:

$$m_{\mathcal{H}}(\epsilon, \delta) \leq m_{\mathcal{H}}^{UC}\left(\frac{\epsilon}{2}, \delta\right) \leq \frac{2}{\epsilon^2} \ln\left(2^{2^{64d}/\delta}\right)$$

However, note that the bound strongly depends on the chosen number representation.

0.3 Bias-complexity tradeoff

Recall the definition of Agnostic PAC Learnability, where if a sufficient number m of training samples are used, for a fixed accuracy ϵ and probability δ, the algorithm will return a hypothesis h that, with a probability $\geq 1-\delta$ over the choice of the m training samples, will satisfy:

$$L_D(h) \leq \min_{h' \in \mathcal{H}} L_D(h') + \epsilon$$

There are still some open (interesting) problems:

- Is an infinite hypotheses class $|\mathcal{H}| = \infty$ PAC learnable?
- How can we choose $\mathcal{H}$ to reduce the first term:

  $$\min_{h' \in \mathcal{H}} L_D(h')$$

  Is there a "universal learner", that is a perfect algorithm A that is able to predict the best $\hat{h}$ for any distribution D?

Given a training set S and a loss function, we'd like to find a function $\hat{h}$ for which $L_d(\hat{h})$ is small. Now consider an algorithm that learns $\hat{h}$ from S. The final choice of $\hat{h}$ will depend on the choice of the set of hypotheses $\mathcal{H}$ and the details of the algorithm.

What if we pick $\mathcal{H}$ extremely large, for example containing all the functions from $\mathcal{X}$ to $\mathcal{Y}$?

This is not going to work, as stated by the following theorem:

Theorem 0.3.1. ***(No-Free Lunch)***.
Let A be any learning algorithm for the task of binary classification with respect to the 0-1 loss over a domain $\mathcal{X}$. Let m be any number smaller than $|\mathcal{X}|/2$, representing a training set size. Then there exists a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ such that:

- *there exists a function $f : \mathcal{X} \to \{0,1\}$ with $L_{\mathcal{D}}(f) = 0$*
- *with probability of at least $1/7$ over the choice of $S \sim \mathcal{D}^m$ we have that $L_{\mathcal{D}}(A(S)) \geq 1/8$.*

Proof: omitted. The main idea is that the training set is smaller than half of the total domain, meaning that the algorithm cannot know anything about what happens in the remaining part. So there exist some target function f that works on the other half in a way that contradicts our estimated labels.

Corollary. Let $\mathcal{X}$ be an infinite domain set and let $\mathcal{H}$ be the set of all functions from $\mathcal{X}$to $\{0,1\}$. Then, $\mathcal{H}$ **is not** PAC learnable.
Proof. We proceed by contradiction. Assume that $\mathcal{H}$ is PAC learnable, meaning that there exists an algorithm A, a function $m_{\mathcal{H}}(\epsilon, \delta)$ and a training set S such that if we choose m elements from S, in such a way that they are i.i.d according to $\mathcal{D}$, for any $\mathcal{D}$, with probability $\geq 1 - \delta$ it holds:

$$L_D(A(S)) < \epsilon$$

(All this works because as $\mathcal{H}$ contains *all the possible functions*, the realizability assumption holds).
Now we choose (arbitrarily) $\epsilon < 1/8$ and $\delta < 1/7$, so that we can apply the No-Free Lunch theorem.
Notice that as $|\mathcal{H}| = \infty$, $|S| < |\mathcal{H}|/2$ ($|\mathcal{X}| > 2m$ in the notation of the theorem). It follows that:

$$L_D(A(S)) \geq \frac{1}{8} \text{ with probability } \geq 1/7$$

which contradicts the PAC thesis:

$$L_D(A(S)) \leq \epsilon \qquad \epsilon < \frac{1}{8}; p \geq 1 - \delta; \delta < \frac{1}{7}$$

This means that for ML algorithms are suited for certain tasks and not others. Also, as taking $\mathcal{H}$ containing all possible functions essentially means assuming no prior knowledge, it is not possible to learn without some previous information or approximation.

So, there are two possibilities for the choice of $\mathcal{H}$:

- $\mathcal{H}$ small: good generalization ($L_D \sim L_S$), but low approximation capabilities (L_S not optimal)
- $\mathcal{H}$ large: good approximation (L_S optimal) but risk of overfitting (L_D may be much larger than L_S)

0.4 Error decomposition

Consider an $\text{ERM}_{\mathcal{H}}$ hypothesis h_S. The true error of $\text{ERM}_{\mathcal{H}}$ can be decomposed as:

$$L_D(h_S) = \min_{h \in \mathcal{H}} L_D(h) + \left[L_D(h_S) - \min_{h \in \mathcal{H}} L_D(h) \right] = \epsilon_{\text{app}} + \epsilon_{\text{est}}$$

- ϵ_{app} is the **approximation error**, that is the minimum true risk achievable by a predictor in $\mathcal{H}$. It depends only on the choice of $\mathcal{H}$, and not of S or the algorithm. It vanishes as $|\mathcal{H}|$ becomes larger. If we assume realizability, then $\epsilon_{\text{app}} = 0$, otherwise it is bounded by the true error of a Bayes Optimal Predictor.
- e_{est} is the **estimation error**, and depends on the non-optimality of the chosen A, and on the choice of the training set S. It usually becomes smaller for larger S, and for smaller $|\mathcal{H}|$, as in a smaller hypothesis class $L_D \sim L_S$.

Note how enlarging $|\mathcal{H}|$ lowers ϵ_{app}, but makes ϵ_{est} larger. So, it is needed to find a *compromise*: the best choice for $\mathcal{H}$ so that the final performance is best.

- If $\mathcal{H}$ is too small, ϵ_{app} is dominant, and this case is denoted as **underfitting**
- If $\mathcal{H}$ is too large, ϵ_{est} becomes high, leading to **overfitting**