Lecture 3: No Free Lunch Theorem

Instructor: Aditya Bhaskara *Scribe: Zhao Chang*

CS 5966/6966: Theory of Machine Learning

January 18th, 2017

Abstract

We will build up towards a characterization of what hypothesis classes are *learnable* in the PAC model. Before getting there, we first show that no learning algorithm can learn *every* function.

1 Review and Introduction

Let us recall some of the notation we have used over the last few lectures.

We have inputs from a space $\mathcal{X}$, and they are assumed to be drawn i.i.d. from an unknown distribution $\mathcal{D}$. We will continue to consider binary classification, and the *target* or *true* hypothesis is given by the function $f : X \mapsto \{0, 1\}$.

Last class, we proved that any *finite* hypothesis class is PAC-learnable, even in the agnostic sense. We denoted by $\mathcal{H}$ a hypothesis class. We also defined the *risk* or the *true loss* of a hypothesis h as

$$\textbf{Risk (Loss) :} \qquad L_{\mathcal{D}}(h) = \Pr_{x\sim\mathcal{D}}[h(x) \neq f(x)].$$

In the agnostic setting, we defined PAC learning as follows.

1.1 definition ((Agnostic) PAC Learning:). A hypothesis class $\mathcal{H}$ is learnable, if $\forall \epsilon, \delta > 0$, there exists an $m = g(\epsilon, \delta)$ and an algorithm A that takes m samples from $\mathcal{D}$ and outputs an $h \in \mathcal{H}$ that satisfies $L_{\mathcal{D}}(h) \leq \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h') + \epsilon$, with probability at least $1 - \delta$.

One key question that we study in this course is the following.

1 Question. What hypothesis classes $\mathcal{H}$ are agnostic PAC learnable?

In the last lecture, we showed that any finite hypothesis class $\mathcal{H}$ is PAC-learnable using $O\left(\frac{\log(|H|)+\log(\frac{1}{\delta})}{\epsilon^2}\right)$ samples.

Our proof in class was only for "consistent" case, but as discussed Chapter 4 of the textbook, the proof also holds in the agnostic case.

As we discussed last time, the dependence on $\mathcal{H}$ is fairly mild – the sample complexity only grows as $\log(|\mathcal{H}|)$.

However, this still does not immediately yield the learnability of *typical* classes $\mathcal{H}$ used in practice – including linear classifiers in $\mathbb{R}^d$, rectangular regions, neural networks, etc.

2 The choice of $\mathcal{H}$ and the *Bias Complexity Tradeoff*

Given some training data, the perpetual question is: what *kind* of a hypothesis do we look for? Or in other words, what $\mathcal{H}$ do we search over?

One options is to be very conservative, and pick only *simple* classes, over which we can optimize and find the best hypothesis efficiently, and with few samples. The issue is that it may not capture the complexities of the problem – for instance when classifying real life images, trying to use a simple linear classifier over the pixels is unlikely to work.

At the other extreme, we can try to be inclusive and use a complex neural network model with a billion parameters. The issue now becomes that we may end up overfitting to the training data. Thus, for the method to work, we may need a huge amount of data (this is partially the reason that neural networks have only now become successful in ML applications). Further, it may become computationally intractable to learn the best such model.

Recall: overfitting is the situation in which we have a tiny error on the training set, but large error on the test, or equivalently, a large true loss $L_{\mathcal{D}}(h)$.

The first issue is called *bias* of the hypothesis class, i.e., the best error $\min_{h\in\mathcal{H}} L_{\mathcal{D}}(h)$ is large. The second is called *complexity*, i.e., the amount of data/time needed to learn the class. This trade-off is a fundamental one in learning.

3 "No Free Lunch" Theorem

The discussion above raises the question: why do we have to fix a hypothesis class when coming up with a learning algorithm? Can we *just learn*? The no-free-lunch theorem formally shows that the answer is NO.

Informal statement: There is no universal (one that works for all $\mathcal{H}$) learning algorithm.

3.1 theorem. *Let A be any learning algorithm for binary classification over $\mathcal{X}$, that uses at most $\frac{|\mathcal{X}|}{2}$ training samples. Then, there exists a distribution $\mathcal{D}$ over $\mathcal{X}$ and a function $h : \mathcal{X} \leftarrow \{0,1\}$ that A "cannot learn". Formally, there exists a h such that the hypothesis h' output by algorithm A after seeing at most $|\mathcal{X}|/2$ examples $(x, h(x))$ satisfies $\Pr_{x\in\mathcal{D}}[h(x) \neq h'(x)] \geq 1/8$, with probability at least $1/8$.*

I.e., if the algorithm A has no idea about $\mathcal{H}$, even the singleton hypothesis class $\mathcal{H} = \{h\}$ (as in the statement of the theorem) is not PAC learnable. The theorem also applies when A is a randomized algorithm. For a full proof, see Chapter 5 of the textbook.

Here, we will only prove a weaker version of the theorem, where we assume that the learner sees only $|\mathcal{X}|/16$ training samples, and we obtain a mildly weaker conclusion that $\Pr[h(x) \neq h'(x)] \geq 1/16$. Further, we assume that A is deterministic. In this case, the proof is a rather simple counting argument.

Proof. For ease of notation, write $n = |\mathcal{X}|$. By saying that the algorithm is deterministic, we mean that given the training samples (and their labels), the h' output by the algorithm is fixed.

The first question we ask is: how many possible inputs are there to A? Every input with m samples is simply a set of tuples $(x_1, h(x_1)), (x_2, h(x_2)), \ldots, (x_m, h(x_m))$. Thus, irrespective of h, the total number of possible inputs to the algorithm is $\binom{n}{m}2^m$, where $m \leq n/16$, by assumption. Now, since A is deterministic, any input to A results in precisely one h'. Thus, any algorithm A can be depicted by a bipartite graph (Figure 1), where the LHS consists of all the possible inputs

of size m, and the RHS consists of all possible hypotheses.

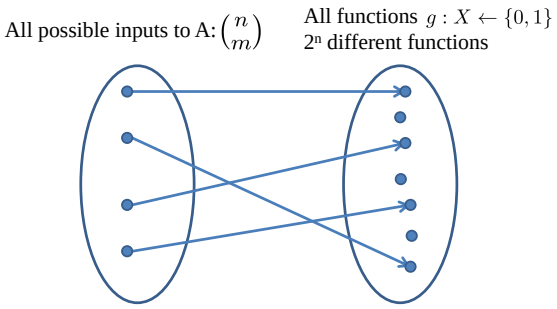

Figure 1: "No free lunch" theorem.

Now the key point to note is that the size of the RHS is 2^n, which a simple calculation reveals, is significantly larger than $\binom{n}{n/16}2^{n/16}$. Thus, *most* of the hypotheses on the RHS will never be the output of A, no matter what the input! Further, we will show that there exists a hypothesis on the RHS that is *not even close* to any possible output of A.

To make this formal, let us say that a $g : \mathcal{X} \mapsto \{0,1\}$ is *mapped* if there exists some input on the LHS upon which A returns g. Let G denote the set of all mapped functions. By the argument above, $|G| \leq \binom{n}{n/16}2^{n/16}$. Now, how many functions $g : \mathcal{X} \mapsto \{0,1\}$ are $1/16$-close to *some* function in G? For any fixed function $g \in G$, the number of f's that are $1/16$-close is $\binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{n/16}$ (as we simply need to pick a subset of the co-ordinates and flip them). This number, from a simple calculation can be seen to be $\leq \binom{n+1}{n/16}$.

A function f is said to be γ-close to a function g (where both functions are from $\mathcal{X}$ to $\{0,1\}$) if they differ in at most γn inputs.

Thus, the total number of f's that are $1/16$-close to *some* function in G is $\leq |G|\binom{n+1}{n/16}$. Using the bound above on $|G|$, we get that the number is at most

$$\binom{n+1}{n/16}\binom{n}{n/16}2^{n/16}.$$

Again an easy calculation yields that this number is $< 2^n$. Thus, there is an h on the RHS that is not $1/16$ close to any of the mapped functions, i.e., it differs from any possible h' output by A with probability at least $1/16$ (we can pick $\mathcal{D}$ to simply be the uniform distribution over $\mathcal{X}$).

For doing these sort of calculations, a useful bound is $\binom{n}{k} \leq \left(\frac{ne}{k}\right)^k$, which is a consequence of Stirling's approximation.

Such an h satisfies the conditions of the theorem. □

4 Extensions

The counting argument used in the proof above yields other interesting statements. For instance, consider the following:

4.1 theorem. *Let $\mathcal{H}$ be a hypothesis class that contains $2^{n/10}$ distinct hypotheses over a domain $\mathcal{X}$ of size n. Then, PAC-learning $\mathcal{H}$ over the uniform distribution over $\mathcal{X}$ requires at least $n/100$ training samples, to achieve an error $< 1/100$, with success probability > 0.1.*

To prove this, consider any deterministic algorithm A that takes fewer than $n/100$ samples. The number of possible inputs is $\binom{n}{n/100}2^{n/100}$, and thus this is the size of G (the set of all hypotheses possibly output by A). The number of hypotheses that are $1/100$-close to these hypotheses can be calculated as

The assumption of deterministic A is not too critical. One of the HW problems will address this.

above, to be roughly $\binom{n+1}{n/100}\binom{n}{n/100}2^{n/100}$, which is smaller than $2^{n/10}$, i.e., the size of $\mathcal{H}$. Thus there exists a hypothesis in $\mathcal{H}$ that is not learnable by any algorithm that takes fewer than $n/100$ samples.

5 Learning Infinite Hypothesis Classes

We have seen that finite classes are learnable. Now, we present a simple example of an infinite-size hypothesis class that is learnable in Figure 2.

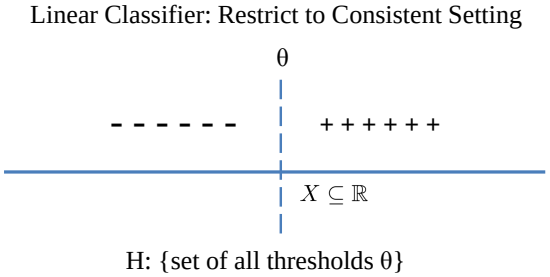

Figure 2: Linear classifier.

Statement: For any ϵ, δ, there exists some $m = g(\epsilon, \delta)$ so that for any distribution D, there exists an algorithm that takes $g(\epsilon, \delta)$ samples and output an classifier with the probability at least $1 - \delta$.

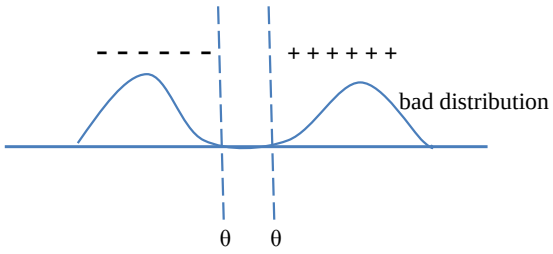

Figure 3: Bad distribution.

Intuitively, as Figure 3 shows, even if we have a bad distribution D and cannot decide what θ is, that doesn't matter. We still have small learning error over that distribution D.