Introduction to Computational Learning Theory **Spring 2021**

Lecture 2: The PAC Model

Lecturer: Roi Livni *Scribe:*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

We now start a formal introduction of the PAC setting introduced in [1], that generalizes the example of learning rectangles in the last class. PAC stands for:

Probably Approximately Correct

2.1 Learning Problem

In the basic statistical setting a learning problem is characterized by the following:

- **Domain Set** An arbitrary set $\mathcal{X}$, this is usually the set of objects that we wish to classify or label. We will usually be concerned with the case where $\mathcal{X} = \mathbb{R}^d$ or $\mathcal{X} = \{0,1\}^d$ for some d: thus, in the general case, each object that the learner might want to classify is described by d real numbers or binary bits. The attributes, or coordinates, of the vector are refered to as *features*.

  As an example, if $\mathcal{X}$ is the set of all 32×32 images, we may describe each image as a set of 32×32 real numbers, each correspond to a pixel in the picture. If for example, we wish to classify a set of students, we might want to describe each student using her grades in a set of d exams.

- **Label Set**: The set $\mathcal{Y} = \{0,1\}$ will describe the labels or class: It is the objective of the learner to assign for each $x \in \mathcal{X}$ a label $y \in \mathcal{Y}$. A slightly more general setup allows us to label x with labels that are not necessarily binary (multi-class or even real numbers). But here we consider a restricted setup where $\mathcal{Y}$ is binary.

- **Hypothesis Class**: An hypothesis class (also *concept class*) consists of target functions (or *predictors*, or *classifiers*) $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ that receive points from the domain $\mathcal{X}$ and return a label from $\mathcal{Y}$.

2.2 Learning algorithm's input/output

Given a learning problem, we analyse the performance of a learning algorithm, which depends on the domain, the label set and the hypothesis class. The learning algorithm has access to the following:

- **Input: Training Data.** The input of a learning algorithm is a finite sample of *labelled examples*: $S = \{(x_1, y_1), \dots (x_m, y_m)\}$. Such labelled examples are also referred to as *training set*. The size of the sample set m is the *sample size*.

> We make an assumption that there exists a distribution D over the domain $\mathcal{X} \times \mathcal{Y}$ that generates the *labelled examples* IID. The algorithm **doesn't know** D (in other words, it doesn't depend on D).

- **Equivalently:** We can also describe a learner as an algorithm that has *Oracle Access to labelled examples*. That means the algorithm can call a procedure EXAMPLE that outputs an example (x, y). We again assume that at every call to EXAMPLE, the output (x, y) is a new independent realization of a random variable $(x, y) \sim D$ distributed by some unknown distribution D. The sample complexity of the algorithm is again defined as the number of calls the algorithm makes to the oracle EXAMPLE
- **Output:** The output of a learning algorithm is a prediction rule which is a function $f : \mathcal{X} \to \mathcal{Y}$ that receives an unlabeled example and outputs a label. f is sometimes called a predictor, or a classifier.

2.3 Expected Error

Given an input sample S, we measure the success of the learner with respect to its success in prediction: Let

$$\ell_{0,1}(h(x), y) = \begin{cases} 1 & h(x) \neq y \\ 0 & \text{else} \end{cases}.$$

The objective of the learner will then be to return the smallest possible error, i.e. to minimize

$$\text{err}_D(h) := \mathbb{E}_{(x,y)\sim D} \left[\ell_{0,1}(h(x), y)\right] \tag{2.1}$$

Eq. 2.1 depicts the *expected error* of the hypothesis h.

Remark. *If no confusion arises, we will omit dependence on D and write $err(h)$.*

2.4 The PAC Model

Let us define the objective of the learner more formally:

Definition 2.1. *[(realizable) PAC Learning] An hypothesis class $\mathcal{H}$ of target functions is PAC learnable if there exists an algorithm A and function $m^A : (0,1)^2 \to \mathbb{N}$ with the following property:*

Assume $S = ((x_1, y_1), \ldots, (x_m, y_m))$ is a sample of IID examples generated by some arbitrary distribution D such that $y = h(x)$ w.p. 1 for some $h \in \mathcal{H}$.

If S is the input of A and $m > m^A(\epsilon, \delta)$ then the algorithm returns a hypothesis h_S^A such that, with probability $1 - \delta$ (over the random choice S) :

$$err\left(h_S^A\right) < \epsilon.$$

The function $m^A(\epsilon, \delta)$ is referred to as the sample complexity *of algorithm A.*

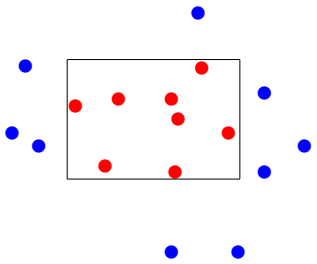
We say that the sample complexity of $\mathcal{H}$ is at most $m_{\mathcal{H}}(\epsilon, \delta)$ if there exists an algorithm A such that $m_{\mathcal{H}}(\epsilon, \delta) \geq m^A(\epsilon, \delta)$ (we will usually omit the subscript $\mathcal{H}$ if there is no room for confusion).

2.4.1 Examples

Example 2.1. *[Axis Aligned Rectangles] The first example of a hypothesis class will be of rectangles aligned to the axis. Here we take the domain $\mathcal{X} = \mathbb{R}^2$ and we let $\mathcal{H}$ include be defined by all rectangles that are aligned to the axis. Namely for every (z_1, z_2, z_3, z_4) consider the following function over the plane*

$$f_{z_1,z_2,z_3,z_4}(x_1, x_2) = \begin{cases} 1 & z_1 \leq x_1 \leq z_2,\ z_3 \leq x_2 \leq z_4 \\ 0 & \text{else} \end{cases}$$

Then $\mathcal{H} = \{f_{z_1,z_2,z_3,z_4} : (z_1, z_2, z_3, z_4) \in \mathbb{R}^4\}$.

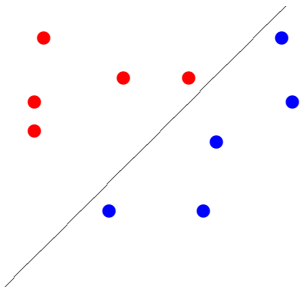


We showed in the last lecture that the sample complexity for learning Axis Aligned Rectangles is

$$m(\epsilon, \delta) \leq \frac{4}{\epsilon} \log 4/\delta.$$

Example 2.2. *[Half-spaces] A second example that is of some importance is defined by hyperplane. Here we let the domain be $\mathcal{X} = \mathbb{R}^d$ for some integer d. For every $\mathbf{w} \in \mathbb{R}^d$, induces a half space by considering all elements $\mathbf{x}$ such that $\mathbf{w} \cdot \mathbf{x} \geq 0$. Thus, we may consider the class of target functions described as follows*

$$\mathcal{H} = \{f_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^d, f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \mathbf{x})\}$$



2.5 Learning Finite Classes

As a first example for learnable classes, we consider finite classes. We assume that $\mathcal{H} = \{h_1, \ldots, h_n\}$ is a finite set of target functions and we denote by $|\mathcal{H}|$ the size of $\mathcal{H}$:

Theorem 2.2 (Finite Classes are learnable)**.** *Any finite hypothesis class is learnable with (realizable) sample complexity*

$$m = O\left(\frac{1}{\epsilon} \ln \frac{|H|}{\delta}\right).$$

The algorithm A that achieves the above rate works as follows: Given a sample S, return $h_S \in \mathcal{H}$ such that:

$$h_S = \arg\min_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \ell_{0,1}(h(x_i), y_i) \tag{2.2}$$

Empirical Risk Minimization (ERM)

Before we set out to prove theorem 2.2 we discuss the suggested algorithm for the solution.

The Empirical Risk Given a sample S the *empirical risk* is defined to be

$$\mathrm{err}_S(h) = \frac{1}{m} \sum_{i=1}^{m} \ell_{0,1}(h(x_i), y_i).$$

The algorithm we described for learning the class $\mathcal{H}$ simply picks any hypothesis h with small empirical error. This is in fact not really an algorithm but more of a meta-algorithm.

ERM Rule: We call an algorithm an Empirical Risk Minimizer (or ERM algorithm, or ERM rule) any algorithm that simply chooses an hypothesis with the smallest empirical risk. In other words, the algorithm solves the optimization problem

$$\underset{h \in \mathcal{H}}{\text{minimize}} \ \mathrm{err}_S(h) = \frac{1}{m} \sum_{i=1}^{m} \ell_{0,1}(h(x), y). \tag{2.3}$$

What we next show is that, in the realizable case, a finite class $\mathcal{H}$ is learnable by *any* ERM rule, or any ERM algorithm. In other words any optimization algorithm for eq. (2.3) will work.

Proof of theorem 2.2

Given a sample S, let h_S^A be the hypothesis returned by algorithm A, and let h^* be the optimal hypothesis in the class. We know by definition that

$$\mathrm{err}_S(h_S^A) \leq \mathrm{err}_S(h^*) = 0.$$

So first, we bound that probability that for some $h \in \mathcal{H}$ with $\text{err}(h) > \epsilon$ we observe a sample S such that $\text{err}_S(h) = 0$. Suppose $\text{err}(h) > \epsilon$:

$$\begin{aligned}
\mathbb{P}(\text{err}_S(h) = 0) &= \mathbb{P}(h(x_1) = y_1 \wedge h(x_2) = y_2 \wedge, \ldots, \wedge h(x_m) = y_m) \\
&= \prod_{i=1}^{m} \mathbb{P}(h(x_i) = y_i) && \text{i.i.d assumption} \\
&\leq \prod_{i=1}^{m} (1-\epsilon) && \text{err}(h) > \epsilon \\
&= (1-\epsilon)^m \\
&\leq e^{-\epsilon m} && (1-\epsilon) \leq e^{-\epsilon}
\end{aligned}$$

Note that our choice m satisfies $m \geq \frac{1}{\epsilon} \ln 1/\delta$. In particular, we have that for a fixed $h \in H$, w.p $1-\delta$

$$\text{err}_S(h) > 0.$$

Since $\text{err}_S(h_S^A) = 0$, does that mean $\text{err}(h_S^A) < \epsilon$, as required?

Note, that to estimate the error of a fixed hypothesis, we relied on the independence between the sample (i.i.d assumption). Notice that h_S^A depends on the sample. Hence the probability of error are not i.i.d any longer.

We will next show that for m sufficiently large, we have (w.p. $(1-\delta)$) that **for every** $h \in \mathcal{H}$ with $\text{err}(h) > \epsilon$, $\text{err}_S(h) \neq 0$. Namely, define:

$$\mathcal{H}_{\text{Sgood}} := \{h \in \mathcal{H} : \text{err}_S(h) = 0\}$$

and

$$H_{\text{good}} = \{h \in \mathcal{H} : \text{err}(h) < \epsilon\},$$

and let us show that w.p. $(1-\delta)$, over the sample S:

$$H_{\text{Sgood}} \subseteq \mathcal{H}_{\text{good}}.$$

This will finish the proof. Indeed, since $h_S^A \in \mathcal{H}_{Sgood}$ we obtain that with probability $(1-\delta)$ over the sample

S, $h_S^A \in \mathcal{H}_{\text{good}}$

To prove the claim we need to show that *no bad hypothesis* (i.e. $h \notin \mathcal{H}_{\text{good}}$) belongs to $\mathcal{H}_{\text{Sgood}}$. So next, we bound the probability that for *some* $h \in H$ we have that $\text{err}_S(h) = 0$ even though $\text{err}(h) > \epsilon$.

$$\begin{aligned}
\mathbb{P}(\{s : \exists h \notin \mathcal{H}_{\text{good}} \wedge \text{err}_S(h) = 0\}) &= \mathbb{P}(\cup_{h \notin \mathcal{H}_{\text{good}}} \{S : \text{err}_S(h) = 0\}) \\
&\leq \sum_{h \notin \mathcal{H}_{\text{good}}} \mathbb{P}(\text{err}_S(h) = 0) && \text{union bound} \\
&\leq (|\mathcal{H}| - |\mathcal{H}_{\text{good}}|)\, |e^{-\epsilon m} \\
&\leq |H| e^{-\epsilon m} \\
&\leq \delta && m > \frac{1}{\epsilon} \ln \frac{|\mathcal{H}|}{\delta}
\end{aligned}$$

2.6 Agnostic Setting

So far we discussed the *realizable* setting and showed that any class is learnable with sample complexity

$$O\left(\frac{\ln |\mathcal{H}|/\delta}{\epsilon}\right).$$

Realizability is often considered a strong assumption. Indeed, in most cases while we can provide a reasonable model for a certain prediction task we don't necessarily assume it is 100% accurate. There are many reasons for this amongst others

- Noisy labels – Data is often manually labeled, there may be mistakes in the labeling process
- Outliers – While we wish to classify typical cases, we should expect that some of the data is non-typical, or even *corrupted*
- Efficiency Vs. Accuracy – In many cases, we might even prefer a *reasonable* model that is *simple* then an accurate model that is expensive (e.g. you want to implement a prediction rule that can be computed fast even at the cost of accuracy.

Let us now consider the following *agnostic* variant of the PAC model

Definition 2.3. *[(agnostic) PAC Learning] A concept class $\mathcal{H}$ of target functions is PAC learnable if there exists an algorithm A and function $m^A : (0,1)^2 \to \mathbb{N}$ with the following property:*

Assume $S = ((x_1, y_1), \ldots, (x_m, y_m))$ is a sample of IID examples generated by some arbitrary distribution D. If S is the input of A and $m > m^A(\epsilon, \delta)$ then the algorithm returns a hypothesis, h_S^A, such that with probability $1 - \delta$ (over the choice of the m training examples):

$$err(h_S^A) < \min_{h \in \mathcal{H}} err(h) + \epsilon$$

The function $m^A(\epsilon, \delta)$ is referred to as the sample complexity *of algorithm A.*

We will prove the following result:

Theorem 2.4 (Finite Classes are learnable – Agnostic case)**.** *Any finite hypothesis class is learnable and any ERM learning rule A learns with sample complexity*

$$m^A = O\left(\frac{\ln |H|/\delta}{\epsilon^2}\right).$$

Similar to the realizable case, we provide a *uniform* bound on our ability to estimate all hypotheses in the calss:

Claim 2.5. *Consider a finite class of target functions $\mathcal{H} = \{h_1, \ldots, h_n\}$ over a domain $\mathcal{X}$. Then if $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ is a sample drawn IID from some arbitrary distribution, and if $m > \frac{2}{\epsilon^2} \ln \frac{2|H|}{\delta}$ then with probability $1 - \delta$ we have that*

$$\forall h \in \mathcal{H},\ |err_S(h) - err(h)| < \epsilon$$

The focus of this lecture will be to prove claim 2.5, before we begin let us show how it implies theorem 2.4.

Proof of theorem 2.4. Let D be some distribution over $\mathcal{X} \times \mathcal{Y}$, and A an ERM rule. Given a sample S, let h_S^A be the hypothesis returned by algorithm A, and let h^* be the optimal hypothesis in the class. We know

by construction that for an ERM rule $\text{err}_S(h_S^A) \leq \text{err}_S(h^*)$. On the other hand choose:

$$m \geq \frac{2}{(\epsilon/2)^2} \ln \frac{2\mathcal{H}}{\delta} = \frac{8}{\epsilon^2} \ln \frac{2\mathcal{H}}{\delta}.$$

we have that with probability at least $(1-\delta)$ that

$$\begin{aligned} |\text{err}_S(h_S^A) - \text{err}(h_S^A))| &\leq \max_{h \in \mathcal{H}} |\text{err}_S(h) - \text{err}(h)| \\ &\leq \frac{\epsilon}{2} \qquad \textit{claim } 2.5 \end{aligned}$$

and also

$$|\text{err}_S(h^*) - \text{err}(h^*)| \leq \frac{\epsilon}{2}.$$

Thus we get that

$$\text{err}(h_S^A) \leq \text{err}_S(h_S^A) + \frac{\epsilon}{2} \leq \text{err}_S(h^*) + \frac{\epsilon}{2} \leq \text{err}(h^*) + \epsilon.$$

□

2.6.1 Proof of Claim 2.5

For the proof we will need the following, well known and fundemental, result about the concentration of IID random variables:

Theorem 2.6 (Hoeffding's inequality)**.** *Let $X_1, \ldots X_m$ be IID random variables such that $0 \leq X \leq 1$. Set $\bar{X} = \frac{1}{m} \sum_{i=1}^m X_i$ then*

$$\mathbb{P}(|\bar{X} - \mathbb{E}(\bar{X})| \geq t) \leq 2e^{-2mt^2}$$

Let $\mathcal{H} = \{h_1, \ldots, h_n\}$. Assume (x, y) are distributed according to some unknown distribution D and let $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ be an IID sample.

For a fixed $h_i \in H$, consider the random variable $\mathrm{err}_S(h_i) = \frac{1}{m}\sum_{j=1}^{m} \ell_{0,1}(h_i(x_j), y_j)$. Note that $\mathrm{err}_S(h_i)$ is the mean of m IID positive random variables bounded by 1 with expectation $\mathrm{err}(h_i)$.

In other words, for a fixed h_i set

$$X_j = \ell_{0,1}(h_i(x_j), y_j).$$

Then $X_1, \ldots, X_m$ are IID, bounded by one, random variables and

$$\bar{X} = \frac{1}{m}\sum_{j=1}^{m} X_j = \frac{1}{m}\sum_{j=1}^{m} \ell_{0,1}(h_i(x_j), y_j) = \mathrm{err}_S(h_i).$$

Also,

$$\begin{aligned}
\mathop{\mathbb{E}}_{S}(\bar{X}) &= \mathop{\mathbb{E}}_{S}[\mathrm{err}_S(h_i)] \\
&= \mathop{\mathbb{E}}_{S}\left[\frac{1}{m}\sum_{j=1}^{m} \ell_{0,1}(h_i(x_j), y_j)\right] \\
&= \frac{1}{m}\sum_{j=1}^{m} \mathop{\mathbb{E}}_{S}[\ell_{0,1}(h_i(x_j), y_j)] \qquad\qquad (\mathop{\mathbb{E}}_{S}[\ell_{0,1}(h_i(x_j), y_j)] = \mathop{\mathbb{E}}_{(x,y)\sim D}[\ell_{0,1}(h_i(x), y)]) \\
&= \frac{1}{m}\sum_{j=1}^{m} \mathrm{err}(h_i) \\
&= \mathrm{err}(h_i)
\end{aligned}$$

Thus, applying Hoeffding's inequality we obtain that for fixed h_i, taking probability over the sample:

$$\mathbb{P}(|\mathrm{err}_S(h_i) - \mathrm{err}(h_i)| > \epsilon) < 2e^{-2m\epsilon^2} \tag{2.4}$$

Now we perform a similar calculation as in previous lecture:

$$\begin{aligned}
\mathbb{P}(\{S :\ \exists h, |\mathrm{err}_S(h) - \mathrm{err}(h)| \geq \epsilon\}) &= \mathbb{P}(\cup_h \{S : |\mathrm{err}_S(h) - \mathrm{err}(h)| \geq \epsilon\}) \\
&\leq \sum_{i=1}^{n} \mathbb{P}(|\mathrm{err}_S(h_i) - \mathrm{err}(h_i)| \geq \epsilon) \qquad \text{union bound} \\
&\leq |\mathcal{H}| \cdot 2e^{-2m\epsilon^2} \qquad m > \frac{1}{2\epsilon^2}\ln\frac{2|\mathcal{H}|}{\delta} \\
&\leq \delta
\end{aligned}$$

The uniform convergence property To prove that a finite class is learnable, we proved a much stronger result. We in fact showed that, given enough samples, we can estimate the expected error of *every* hypothesis in the class, even the hypotheses our learner doesn't choose.

Definition 2.7 (Uniform Convergence Property)**.** *We say that a hypothesis class $\mathcal{H}$ has the* uniform convergence property *if there exists a function $m : (0,1)^2 \to \mathbb{N}$ such that for every $\epsilon, \delta \in (0,1)$ and for every probability distribution D over $\mathcal{X}$, if $S = ((x_1, y_1), \ldots, (x_m, y_m))$ is a sample of size $m \geq m(\epsilon, \delta)$ drawn IID according to D then with probability at least $(1-\delta)$ we have that*

$$\forall h \in \mathcal{H}, \quad |err_S(h) - err(h)| < \epsilon$$

References

[1] Leslie G Valiant. A theory of the learnable. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 436–445. ACM, 1984.