COS-511: Learning Theory **Spring 2017**

Lecture 1: Brief Overview – PAC Learning

Lecturer: Roi Livni

We begin with a brief overview of the classical results in the Statistical Model of Binary Classification (aka PAC). The PAC model is an extremely attractive model for learning. As we will discuss in the next few lectures, we can fairly well characterize what it means to be *learnable* in this model as well as characterize successful learning algorithms.

Much of the rest of the course (at least the first half) will deal with the extensions and the analysis of more realistic models: as we will see, while some of the results from PAC can be extended to other domains, alot of the theory becomes much more involved once we extend the model.

We refer to to [**?**] for further reading. This section is aligned with the first few chapters in this book.

1.1 Statistical Learning

1.1.1 Learning Problem

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

- **Domain Set** An arbitrary set χ, this is usually the set of objects that we wish to classify or label. We will usually be concerned with the case where $\chi = \mathbb{R}^d$ or $\chi = \{0,1\}^d$ 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 χ 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 $\mathcal{Y}$ will describe the labels or class: It is the objective of the learner to assign for each $x \in \chi$ a label $y \in \mathcal{Y}$. Perhaps the simplest learning task is when the labels are binary, i.e $\mathcal{Y} \in \{0,1\}$ or $\mathcal{Y} \in \{-1,1\}$.

- **Concept Class**: A concept class also consists of target functions $\mathcal{C} = \{h : h : \chi \to \mathcal{Y}\}$ the receives points from the domain χ and returns a labeling from $\mathcal{Y}$.

We make an assumption that there exists a distribution D over the domain $\chi \times \mathcal{Y}$ that generates *labeled examples* i.e. pairs $(x, y) \in \chi \times \mathcal{Y}$.

1.1.2 Learning algorithm's input/output

Given a learning problem, we analyse the performance of a learning algorithm which has access (besides to the domain and labels) to the following:

- **Input: Training Data.** $S = \{(x_1, y_1), \dots (x_m, y_m)\}$ is a finite set of pairs in $\chi \times \mathcal{Y}$. This is the *input* that the learner has access to. Such labeled examples are also referred to as *training examples* or *labeled sample set*. The size of the sample set m is the *sample size*. We will generally assume that the sample S was generated by drawing m IID samples from the distribution D.
- **Output: Hypothesis.** A Hypothesis class consists of a subset of target functions $\mathcal{H} = \{h : h : \chi \to \mathcal{Y}\}$ that turns unlabeled samples to labels. Each learning algorithm outputs a hypothesis, the class of hypotheses the learner may return is the algorithms hypothesis class.

> We stress that we **do not** assume that the learner has access to the distribution D. i.e. the hypothesis outputted by the learner is a function of S alone.

1.1.3 Generalization Error

Given an input S, the learner is required to output a hypothesis $h : \chi \to \mathcal{Y}$. This functions is called *predictor*, or a *classifier*.

Clearly, the objective is not to return an arbitrary predictor and we measure the success of the learner with respect to some loss function $\ell : \chi \times Y \to \mathbb{R}$. Our focus in binary classification is w.r.t the $0-1$ loss function defined as follows

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

In words, a learner suffers loss 1 on every example on which it is wrong, and suffers no loss whenever it is correct. The objective of the learner is then to return the smallest possible error, i.e. to minimize

$$\text{err}(h) := \mathbb{E}_{(x,y)\times D}\left[\ell_{0,1}(h(x), y)\right] \tag{1.1}$$

Eq. 1.1 depicts the *generalization error* of the hypothesis h.

We next, formally describe how we measure the success of a learning algorithm, within the PAC framework:

1.2 The PAC Model

We loosely defined the objective of the learner to minimize its generalization error. To assume that one can choose the exact optimal classifier h by observing a finite sample set may be too strong. In the PAC (Probably Approximately Correct) model, introduced by Valiant at [**?**] we measure the success of the learner if it can (with enough sample size) return an approximately accurate solution with high probability. The general aim of a learning algorithm, is to outperform some prespecified class of function, referred to as a *concept class*.

Thus, learning is loosely defined as follows: A learning algorithm A receives m IID examples drawn from a distribution D over $\chi \times \mathcal{Y}$. We assume that the labels are consistent with some concept $h \in \mathcal{C}$ that belongs to some concept class we wish to learn (this assumption will then be relaxed). The algorithm observes the

sample and chooses a hypothesis $h \in \mathcal{H}$ from some hypothesis class. The aim of the algortihm is to return a hypothesis with "small" error. Formally we define PAC learning as follows:

Definition 1.1 ((realizable) PAC Learning)**.** *A concept class $\mathcal{C}$ of target functions is PAC learnable (w.r.t to $\mathcal{H}$) if there exists an algorithm A and function $m_{\mathcal{C}}^{A} : (0,1)^2 \to \mathbb{N}$ 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 $y_i = h(x_i)$ for some $h \in C$ almost surely. If S is the input of A and $m > m_{\mathcal{C}}^{A}(\epsilon, \delta)$ then the algorithm returns a hypothesis $h_S^A \in \mathcal{H}$ such that, with probability $1 - \delta$ (over the choice of the m training examples):

$$err(h_S^A) < \epsilon$$

The function $m_{\mathcal{C}}^{A}(\epsilon, \delta)$ is referred to as the sample complexity *of algorithm A.*