

PAC Learning

Nader H. Bshouty

Eyal Kushilevitz

Abstract

Here we define the PAC learning model.

1 Learning from Examples

In this chapter we discuss the model of *learning from examples*. This model represents situations in which a learning algorithm can only view random examples (that is, pairs of the form $(x, f(x))$) and, based on these examples, should succeed to learn the target function $f : \{0, 1\}^n \rightarrow \{0, 1\}$. The first question that we have to ask is what we will consider a “success”. Note that the examples that the learning algorithm sees are “random”. Even though we still have not defined what does “random” mean, it is clear that if the learning algorithm is unlucky these examples may not contain enough information. An extreme case would be where all examples are just the same x . A less extreme case would be if all the examples are, say, n -bit strings that start with a prefix “001”. It is therefore clear that if we wish to define a model of learning from (random) examples, a crucial point is to formulate “correctly” the notion of success. In particular, this notion should be “reasonable” in the sense that there is a chance that we will find learning algorithms that meet it. Informally, we will ask the algorithm to be *Probably Approximately Correct (PAC)*. That is,

- We allow a small probability that the learning algorithm fails. Intuitively, this probability captures the chances that the examples seen by the algorithm do not contain enough information to learn.
- We allow the learning algorithm not to learn the target function *exactly* but rather to approximate it. That is, finding a function h that is “close” to f .

In order to give meaning to the above notions, we define a probability distribution \mathcal{D} on the domain of the function, $\{0, 1\}^n$. Therefore, when we say a “random” example we mean a pair $(x, f(x))$, where x is chosen according to the distribution \mathcal{D} . Moreover, we assume that if the learning algorithm sees several examples then they are chosen independently of each other. The probability distribution also allows us to define how well a function f approximates the target function f :

$$\text{error}(h, f) \triangleq \Pr_{x \in \mathcal{D}} (h(x) \neq f(x)).$$

In figure 1 the domain of inputs $\{0,1\}^n$ is drawn. Inside this space, the functions f and h are drawn as rectangles (representing the set of “1”s of each function). Then, the grey area represents the points in the domain on which f and h disagree. The probability of these points is exactly $\text{error}(h, f)$.

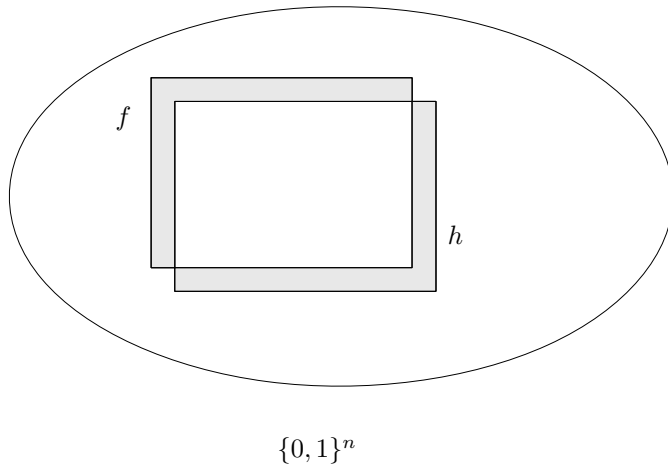


Figure 1: $\text{error}(h, f)$

The value of $\text{error}(h, f)$ is the probability that, if we pick a new random example x , according to \mathcal{D} , the function h “disagrees” with f (i.e., it classifies x incorrectly). The quality of success will be measured by two parameters ε and δ , where ε will be the desired quality of approximation (i.e., the learning algorithm tries to find h such that $\text{error}(h, f) \leq \varepsilon$) and δ is the failure probability (i.e., the learning algorithm is required with probability at least $1 - \delta$ to find such a function h). This means that the error probability is measured with respect to the same distribution according to which the random examples are chosen. Again, this is unavoidable since if, for example, the learning algorithm will get random examples from a distribution which provides only examples with first bit 0 and the error will be measured with respect to distribution on strings whose first bit is 1 then clearly the learning algorithm has no chance to do much.

In the next section we present a formal definition of the *PAC model* which is based on the intuition provided here.

2 The Model

We are now ready for the formal definition of the model.

Definition 2.0.1 *Let \mathcal{C} be a class of boolean functions $f : \{0,1\}^n \rightarrow \{0,1\}$. We say that \mathcal{C} is (efficiently) PAC-learnable if there exists an algorithm \mathcal{A} such that:*

for every $f \in \mathcal{C}$
 for every probability distribution \mathcal{D}
 for every $0 < \varepsilon < 1/2$
 and for every $0 < \delta < 1$
 algorithm \mathcal{A} on input ε and δ and a source of random examples, distributed according to the distribution \mathcal{D} , runs in time polynomial in $1/\varepsilon$, $1/\delta$ and in the relevant parameters of the class \mathcal{C} (in particular it gets at most polynomially-many examples), and with probability at least $1 - \delta$ halts with a (polynomial-time computable) function h such that $\text{error}(h, f) \leq \varepsilon$.

Before showing how to use this definition let us make some remarks that clarify why this is a good definition. First, as discussed above, $\varepsilon = 0$ is impossible to get (for a concrete example, see exercise below). On the other hand, $\varepsilon = 1/2$ is useless since essentially the same effect can be achieved by tossing a coin. As for the choice of δ , again $\delta = 0$ is impossible while $\delta = 1$ is trivial. Note that allowing the complexity to depend on $1/\varepsilon$ and $1/\delta$. Intuitively, if we wish to get a better quality of learning we have to pay for this. Moreover, if there was no dependency on $1/\varepsilon$ and $1/\delta$ we were able to pick them so small that we will be again in the same impossible cases discussed above.

Next, note that the requirement is with respect to *all* distributions. Moreover, we ask for a single algorithm \mathcal{A} for all distributions (and not that for every distribution \mathcal{D} there exists an algorithm that was designed for the specific distribution \mathcal{D}). Intuitively, this means that algorithm \mathcal{A} “does not know” the distribution. We stress however that the distribution is only on the domain and that, for example, there is no probability distribution associated with the target functions (models that do make such assumptions are called *Bayesian* models and are not discussed in this book).

Exercise 2.0.1 Consider the class \mathcal{C} of all singleton functions (i.e., the functions f_a such that $a \in \{0, 1\}^n$ and $f_a(a) = 1$ but $f_a(b) = 0$ for all $b \neq a$).

- Prove that this class cannot be learned in polynomial time with $\varepsilon = 0$.
 Hint: Fix δ to some constant and take \mathcal{D} to be the uniform distribution.
- Give a PAC-learning algorithm for this class.