Meta Complexity

Lecture 3

Ronald de Haan
me@ronalddehaan.eu

University of Amsterdam

June 3, 2025

What will we cover in this lecture?

- Natural proofs
- Learning

Natural proofs

- Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function and let $c \geq 1$.
- Any proof that f does not have n^c-sized circuits can be viewed as exhibiting some property that f has, and which every function with an n^c-sized circuit does not have.
- That is, such a proof can be viewed as providing a predicate $\mathcal{P}$ on Boolean functions such that $\mathcal{P}(f) = 1$ and:

$$\mathcal{P}(g) = 0 \quad \text{for every } g \in \mathsf{SIZE}(n^c) \qquad (n^c\text{-usefulness})$$

Natural proofs (ct'd)

- We say that such a predicate $\mathcal{P}$ is *natural* if in addition to n^c-usefulness it satisfies the following two conditions:
  - Constructiveness: There is a $2^{O(n)}$-time algorithm that on input (the truth table of) a function $g : \{0,1\}^n \to \{0,1\}$ outputs $\mathcal{P}(g)$—i.e., a polynomial-time algorithm.
  - Largeness: The probability that a random function $g : \{0,1\}^n \to \{0,1\}$ satisfies $\mathcal{P}(g) = 1$ is at least $1/n$.

Natural proofs (ct'd)

Theorem (Razborov-Rudich 1997)

Suppose that subexponentially strong[1] one-way functions exist.

Then there exists a constant c such that there is no n^c-useful natural property $\mathcal{P}$.

[1] A OWF is called *subexponentially strong* if it resists inverting even by a 2^{n^ϵ}-time adversary, for some fixed $\epsilon > 0$.

MCSP $\in$ P gives natural properties

- If MCSP $\in$ P, then there exists a n^c-useful natural property $\mathcal{P}$ for each $c \geq 1$:
  - Take the property $\mathcal{P}$ of not having circuits of size $\leq n^c$.
  - By definition, this is n^c-useful.
  - Because MCSP $\in$ P, the property $\mathcal{P}$ is constructive.
  - It is also large, because there are 2^{2^n} functions $g : \{0,1\}^n \to \{0,1\}$ and there are only $\leq 2^{O(n^c \log n^c)}$ circuits of size $\leq n^c$.

Probably Approximately Correct (PAC) learning

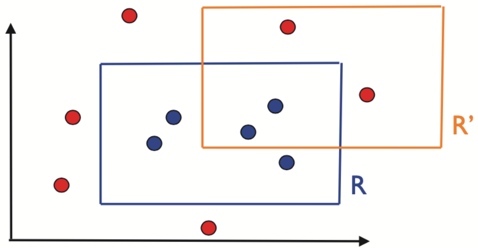
- PAC-learnability is a theoretical notion of what is learnable
- Basic setup:
  - A set $\mathcal{X}$ of *examples*
  - A set $\mathcal{Y}$ of *labels*
  - A *concept* is a function $c : \mathcal{X} \to \mathcal{Y}$
  - A *concept class* C is a set of concepts
  - Examples are drawn independently and identically distributed (i.i.d.) according to some *distribution* D

The learning problem

- The learning problem is to:
  - learn a concept $c \in C$,
  - based on some samples $x_i \in \mathcal{X}$, together with their correct label $c(x_i) \in \mathcal{Y}$
  - and to learn it *approximately correctly*.
- Ingredients:
  - You're **not** given the concept c.
  - Based on a *hypothesis set* H. (a set of concepts)
  - You're given some samples $S = (x_1, \ldots, x_m)$ together with their labels $(c(x_1), \ldots, c(x_m))$, drawn according to the distribution D.
  - Goal: select some $h_S \in H$ that has a small enough error $R(h)$:

$$R(h) = \Pr_{x \sim D} [h(x) \neq c(x)]$$

Example: learning a rectangle



- $\mathcal{X} = \mathbb{R}^2$
- $\mathcal{Y} = \{0, 1\}$
- $C = H =$ "all rectangles in $\mathbb{R}^2$"
- D is any distribution over $\mathbb{R}^2$
- Blue dots indicate examples with label 1,
  red dots indicate examples with label 0
- R is the concept c that is to be learned, and R' is a hypothesis

Definition of PAC-learnable

- A concept class C is PAC-learnable using the hypothesis class H if:
  - there exists a probabilistic algorithm $\mathcal{A}$ and a polynomial p,
  - such that for all $\epsilon > 0$ and $\delta > 0$, for all distributions D on $\mathcal{X}$, and for each concept $c \in C$,
  - for each $m \geq p(\frac{1}{\epsilon}, \frac{1}{\delta}, n, \text{size}(c))$ it holds that:

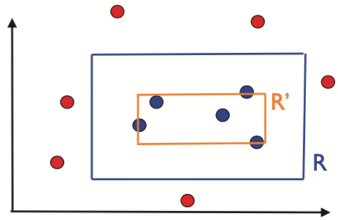
$$\Pr_{S \sim D^m}[R(h_S) \leq \epsilon] \geq 1 - \delta,$$

    where $h_S \in H$ is the hypothesis computed by algorithm $\mathcal{A}$ when given samples S,

    where n is the size needed to represent an element $x \in \mathcal{X}$,
    and where $\text{size}(c)$ is the size needed to represent the concept c.

- If $\mathcal{A}$ also runs in time $p(\frac{1}{\epsilon}, \frac{1}{\delta}, n, \text{size}(c))$, then C is *efficiently PAC-learnable*.

Back to the example



- A PAC-learning algorithm $\mathcal{A}$ for this example problem is one that does the following:
  - Given a large enough sample S of examples
  - outputs the smallest rectangle R' that contains all sampled examples with label 1 (all blue dots)
- One can prove that for any D and any $\epsilon > 0$, $\delta > 0$, this algorithm outputs (with probability $\geq 1 - \delta$) a hypothesis h that has error $R(h) \leq \epsilon$ (for distribution D)

Example of something not PAC-learnable, unless RP = NP

- Learning problem that allows us to solve Dominating Set:
- Given a graph $G = (V, E)$ with vertices $\{v_1, \ldots, v_n\}$, and some $k \in \mathbb{N}$, construct:
  - $\mathcal{X} = \{0,1\}^n$, $\mathcal{Y} = \{0,1\}$
  - C corresponds to all subsets of V
    - For $c_S \in C$ and $x = (x_1, \ldots, x_n) \in \mathcal{X}$,
      $c_S(x) = 1$ if and only if for some $v_i \in S$ it holds that $x_i = 1$.
  - H corresponds to all subsets of V of size k
- You can use a PAC-learning algorithm $\mathcal{A}$ for this problem to make a probabilistic algorithm for Dominating Set:
  - Idea: set ϵ to the right value $(\frac{1}{2n})$, and feed $\mathcal{A}$ n samples that correspond to the input graph G
    (a sample for each vertex v_i, where $x_j = 1$ iff $i = j$ or $\{i, j\} \in E$)

PAC learning with membership queries

- In the "regular" PAC learning setting, the learning algorithm only gets access to samples $(x, c(x))$ drawn from the distribution $\mathcal{D}$.

  That is, the algorithm $\mathcal{A}$ is not allowed to choose which inputs to get a correct answer for.

- In PAC learning with membership queries, the learning algorithm $\mathcal{A}$ has access to an oracle that, given any x, produces $c(x)$.

Summary

- Natural proofs
- Learning