TTIC 31290: Machine Learning for Algorithm Design (Fall 2025)
Avrim Blum and Dravyansh Sharma

Lecture 5: 10/14/25 Lecturer: Dravyansh Sharma

Outline for today

- Motivation for data-driven algorithm design (slides)
- Setup
- PAC Learning
- Algorithm Design as PAC Learning

1 Setup

We use X to denote the space of (set of all possible) *examples* (or *instances*). In the classical learning theory set up, these examples correspond to objects like images or text that need to be classified by the machine learning model. For algorithm design, this will be the space of input instances to the algorithm (e.g. all possible clustering instances).

A simple learning setting is *binary classification*, where each example has a binary *label* that we want to predict. We might want to predict whether there is a cat in an image, or whether an email is a spam. It is natural to think of our model as a mapping from the examples in X to binary *labels*. We formalize this using functions $h : X \to \{0, 1\}$ called *hypotheses*. We assume we have a set of such functions H, the *hypothesis class*. Intuitively, the goal of learning would be to find a good mapping from among the functions in H.

We can also think of the true mapping as a binary function $c^* : X \to \{0, 1\}$, called the *target concept*, which gives the correct label for each $x \in X$. We can similarly define a *concept class* C as a set of concepts that contain c^*.

2 PAC Learning

A learner receives a sample

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\},$$

where each x_i is drawn i.i.d. from an unknown distribution D over X, and $y_i = c^*(x_i)$ for some unknown target concept $c^* \in C$. The learner outputs a hypothesis $h \in H$.

The *true error* of a hypothesis h with respect to D and c^* is

$$\text{err}_D(h) = \Pr_{x \sim D}[h(x) \neq c^*(x)].$$

We say that the concept class C is *PAC learnable* (Probably Approximately Correct learnable) using hypotheses in H if there exists a polynomial function $m_C(\epsilon, \delta)$ and a learner such that for all

$\epsilon, \delta \in (0,1)$, for all distributions D on X, and for all target concepts $c^* \in C$, when the learner is given $m \geq m_C(\epsilon, \delta)$ i.i.d. labeled examples from D, it outputs $h \in H$ satisfying

$$\Pr\left[\text{err}_D(h) \leq \epsilon\right] \geq 1 - \delta.$$

If we require $H = C$, then this is typically called "proper PAC learning".

Can we achieve PAC learning? What sample size $m(\epsilon, \delta)$ is sufficient to guarantee (ϵ, δ)-PAC learning?

Turns out the following simple learner achieves PAC learning for finite concept classes.

Definition 1. *(Consistent learner.) Define the sample error of hypothesis $h \in H$ on sample $S = (x_1, \ldots, x_m)$ as its average error over the sample $\text{err}_S(h) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}[h(x_i) \neq c^*(x_i)]$. A consistent learner simply outputs any $h \in H$ with zero sample error (if one exists).*

Since we assumed $C = H$, we have $\text{err}_S(c^*) = 0$, and the consistent learner will always output a hypothesis in this case.

Theorem 1. *The consistent learner PAC learns any finite concept class C, and needs only*

$$\frac{1}{\epsilon}\left(\ln(|C|) + \ln\frac{1}{\delta}\right)$$

examples to output a hypothesis of error at most ϵ with probability at least $1 - \delta$.

Proof. We define a set of *bad* hypotheses as follows.

$$B = \{h \in C \text{ with } \text{err}_D(h) > \epsilon\}$$

Fix a bad hypothesis $h \in B$. The probability that h makes an error on any $x \in D$ is at least ϵ. Therefore, the probability that this hypothesis is consistent with m examples in the sample is at most $(1 - \epsilon)^m$.

So, by union bound, the probability that there exists a bad hypothesis consistent with the sample S is at most $|B|(1 - \epsilon)^m \leq |C|(1 - \epsilon)^m$. To get the desired result, we simply set this to the failure probability δ (that is, we fail to learn a hypothesis with error at most ϵ in this event) and solve for m. □

3 Algorithm Design as PAC Learning

We will now give a formulation for algorithm design as PAC Learning.

We denote by Π the set of problem instances of interest for a fixed problem of interest (e.g. clustering or vertex cover). We also fix $\mathcal{A}$ a (potentially infinite) family of algorithms. We also fix a utility function $u : \Pi \times \mathcal{A} \to [0, U]$, where $u(x, A)$ measures the performance of the algorithm A on problem instance $x \in \Pi$. For example, u could denote the algorithm's running time and H could be the time-out deadline.

The "domain-specific information" is modeled by the unknown input distribution D. The learner is given m i.i.d. samples $x_1, \ldots, x_m \in \Pi$ from D, and (perhaps implicitly) the corresponding performance $u(x, A)$ of each algorithm $A \in \mathcal{A}$ on each input x_i. The learner uses these samples to suggest an algorithm $\hat{A} \in \mathcal{A}$ to use on future inputs drawn from D. We seek learners that almost always output an algorithm of $\mathcal{A}$ that performs almost as well as the optimal algorithm A^* for D that maximizes $\mathbb{E}_{x\sim D}[u(x, A)]$ over $A \in \mathcal{A}$. Phrased in the PAC terminology, we would like to guarantee that for any $\epsilon, \delta \in (0, 1)$, for every distribution D over Π, given $m \geq m_{\mathcal{A}}(\epsilon, \delta)$ i.i.d. samples from D, we find an algorithm $\hat{A} \in \mathcal{A}$ such that,

$$\Pr\left[(\max_{A\in\mathcal{A}} \mathbb{E}_{x\sim D}[u(x, A)]) - \mathbb{E}_{x\sim D}[u(x, \hat{A})] \leq \epsilon\right] \geq 1 - \delta.$$

Additional Resources

- Maria-Florina Balcan, "Data-Driven Algorithm Design" (book chapter). In *Beyond Worst Case Analysis of Algorithms*, Tim Roughgarden (Ed). Cambridge University Press, 2020. `https://www.cs.cmu.edu/%7Eninamf/papers/data-driven-chapter.pdf`
- Maria-Florina Balcan, Lecture 1 for 10-806 "Foundations of Machine Learning and Data Science". `https://www.cs.cmu.edu/~ninamf/courses/806/lect-09-09.pdf`