Lecture 14:

PAC learnability

Uniform convergence
Recall: Probably Approximately Correct (PAC)

$x$: instance space set of all instances (input space / domain)

$c: x \rightarrow \{+1,-1\}$ concept a function to label elements

$C$: concept class a collection of labeling functions

$c^*$: target concept $c^* \in C$ and label all instances correctly

$D$: target distribution distribution over instances

Sample / training dataset: $\langle x_1, c^*(x_1) \rangle$

$\langle x_2, c^*(x_2) \rangle$

$\vdots$

$\langle x_n, c^*(x_n) \rangle$
Recall

**PAC learning (Probably Approximately Correct)**

Suppose that we have a concept class $C$ over $X$. We say that $C$ is PAC learnable if there exists an algorithm $A$ s.t.

$$A \in C, \forall D \text{ over } X, \forall \varepsilon, \delta \in (0, 0.5]$$

$A$ receives $\varepsilon, \delta$, and samples $<x_1, c(x_1)>$ ...

$$<x_n, c(x_n)>$$

where $x_i$'s are iid samples from $D$.

Then, w. p. $\geq 1-\delta$, $A$ outputs $\hat{C}$ s.t.

$$\text{err} \left( \hat{C} \right) \leq \varepsilon.$$

The probability is taken over the randomness in the samples and any internal coin flips of $A$. 

other notation

true error:

\[
\text{err } (c) = \Pr \left[ \text{c}(\mathbf{x}) \neq y \right] \quad \text{for } (\mathbf{x}, y) \sim \mathcal{D}
\]

training error:

\[
\text{err } (c) = \frac{\text{# Samples in } T}{\text{samples in } T} \quad \text{such that } \text{c}(\mathbf{x}; i) \neq y_i
\]

fraction of samples in the training set that \( c \) is mis-labeled.
Example 2: Boolean conjunctions:

\[ X = \{0, 1\}^n \]

literals \( x_i, \bar{x}_i \)

Conjunction = \( \{ \text{literal} \} \)

\[ \text{l literal \( \land \) conjunction} \]

Concept: a conjunction

Example: \( h(x) = x_1 \land \bar{x}_2 \)

\( h((1, 0, 1)) = 1 \)

\( h((0, 0, 1)) = 0 \)

\( \mathbb{H} \): the set of all conjunction functions

Goal: PAC learning of \( \mathbb{H} \)

Suppose we have samples of the form \( \langle x, h^*(x) \rangle \) from a distribution \( D \)

\( \Rightarrow \) realizable
Algorithm:
- start with $h = \lnot x_1 \lor x_2 \lor x_3 \lor \ldots \lor x_n$ 
- Try $m = ?$ examples
  - ignore negative example.
  - for positive example, remove inconsistent terms.
- Output $h$

Deleting an inconsistent literal

$h = \sqrt{x_2 \land x_3 \land \overline{x_4}}$

Sample: $< (1, 0, 1, 0), 1 > \rightarrow x_2$ is inconsistent

$\downarrow$

we delete $x_2$ from $h$

$\downarrow$

new $h = x_3 \land \overline{x_4}$
Our goal is to analyze the performance of the algorithm.

First, we start by the error of the output hypothesis \( \hat{h} \).

Initially, \( h \) contains all literals. We only remove inconsistent literals. So, we never removes literals in \( h^* \) from \( h \). That is, \( \hat{h} \) contains all the literals in \( h^* \). This fact implies if \( h^*(x) = 0 \), \( \hat{h}(x) \) must be zero too.

\[ \Rightarrow \text{Hence } \hat{h} \text{ always labels } x \text{ correctly if } h^*(x) \geq 0 \]
Now consider the rest of the domain elements \( x \) such that \( h^*(x) = 1 \). If \( \hat{h} \) makes a mistake (i.e. \( \hat{h}(x) = 0 \)), there must be a literal in \( \hat{h} \), \( z \) that is inconsistent:

\[
\text{true error of } h = \text{err}(h) = \text{Pr} \left[ \hat{h}(x) \neq h^*(x) \right] = \text{Pr} \left[ \exists z \in \hat{h} \text{ a literal such that } x_{12} = 0 \text{ but } h^*(x) = 1 \right] = \sum_{z \in \hat{h}} \text{Pr} \left[ \exists x_{12} = 0 \text{ but } h^*(x) = 1 \right]
\]

by the union bound

Call this \( \rho(z) \)

\[
\sum_{z \in \hat{h}} \rho(z) \star
\]
We call a literal bad iff $p(z)$ is at most $\frac{\varepsilon}{2^n}$.

$$\text{bad } z \iff p(z) > \frac{\varepsilon}{2^n}$$

Using * it is easy to see if no bad literal survives in $h$ then

$$\text{err } (h) \leq \sum_{z \in \hat{h}} p(z) \leq 2n \cdot \frac{\varepsilon}{2^n}$$

$\Rightarrow$ hence the error of $\hat{h}$ is good
Now, let's focus on the probability of $\Pr(\hat{h}) \in \\ \Pr[\text{outputting an inaccurate } \hat{h}] \\
\Pr[\text{training set}] = \Pr[\text{err}(\hat{h}) > \varepsilon] \\
\leq \Pr[\exists \text{ a bad literal } z \text{ in } \hat{h}] \\
\leq 2n \cdot \Pr[\text{a bad literal survives all the m samples (not been deleted) }] \\
\leq 2n \cdot (1 - p(2))^m \leftarrow \text{It is not hard to see that with probability } p(2) \\
\leq 2n \cdot (1 - \frac{\varepsilon}{2n})^m \\
\leq 2n \cdot e^{-\frac{\varepsilon m}{2n}} \leq 8 \\
\uparrow \\
\text{we delete } z \text{ at every round.}
by setting \( m = \frac{2n}{\varepsilon} \log \left( \frac{2n}{\delta} \right) \)

Hence our algorithm with prob. 1 - \( \delta \) output \( \hat{h} \) that has low error, \( (err(h) \leq \varepsilon) \)

\( \Rightarrow \) we PAC-learned \( \ddagger \) :)
**ERM**

In both example we picked concepts $\hat{R}$ and $\hat{h}$ that were consistent with the samples in the training set.

What we did is called:

**ERM**: Empirical Risk Minimization

\[ \text{comes from samples error} \]

**ERM algorithm**: it finds a concept $\hat{h}$ such that $\text{err}(\hat{h}) = 0$
Uniform convergence. (UC)

Class $C$ has the uniform convergence property if $\forall \varepsilon, \delta \in (0, 1)$, $\exists m$ (as a function of $\varepsilon, \delta, \Pi$, but not $D$ since we don't know $D$), s.t. for a training set of size $m$:

$$\Pr_{T \sim D^m} \left[ \forall c \in C : |\hat{\text{err}}_T(c) - \text{err}(c)| \leq \varepsilon \right] \geq 1 - \delta$$

Uniform convergence implies agnostic PAC learnability via EMR.

$UC \Rightarrow \forall c \in C_B \quad \hat{\text{err}}_S(c) \geq \text{OPT} + \varepsilon/2$

$UC \Rightarrow c^* = \text{the best option} \quad \hat{\text{err}}_S(c^*) \leq \text{OPT} + \varepsilon$

BAD

\[
\begin{align*}
0 & \quad \text{OPT} \quad \text{OPT} + \varepsilon \quad \text{BAD} \\
\text{OPT} + \varepsilon/2 & \quad \text{error}
\end{align*}
\]
There are two types of error in the agnostic setting:

\[ \text{err}(\hat{c}) < \min_{c \in C} \text{err}(c) + \varepsilon \]

- \( \varepsilon_{\text{est}} \) = estimation error
- \( \varepsilon_{\text{app}} \) = approximation error

depends only to the choice of the class \( C \)

Is \( C \) rich enough to capture how data is labeled?

- \( C \) larger: \( \varepsilon_{\text{app}} \) more complex
- \( \varepsilon_{\text{est}} \)