Lecture 21 (Part I)

average vs. worst case complexity

Last time:

Weak vs. Strong Learning

Def. Algorithm A "weakly PAC learns" concept class C if $\exists \gamma > 0$

s.t. $\forall c \in C$ + $\forall$ dists $\mathcal{D}$

$\forall \delta > 0$ ← ($\delta = \frac{1}{4}$ or $\frac{1}{n^2}$ doesn't affect)

with prob $\geq 1 - \delta$

given examples of c (labelled)

A outputs h s.t. $\Pr_{\mathcal{D}}[h(x) = c(x)] \geq \frac{1}{2} + \gamma$

in "strong" learning this is $1-\varepsilon$

not good compared to $1-\varepsilon$ or 99%

↑ advantage over guessing

Assume $c \in C$ maps $X \to \{0,1\}$

↖ domain

$\mathcal{C}$ ← parametrized by "size" n of inputs

$\mathcal{C}$ is concept class where s ← fctn of n bounds size of WL-description of concepts

Thm $\mathcal{C}$ weakly learnable $\Rightarrow \exists$ efficient algorithm

Strong learner:

- using $\frac{\text{poly}(n, s, \log 1/\varepsilon, \log 1/\delta)}{\varepsilon}$ samples & time
- outputs hypotheses of size $\text{poly}(n, s, \log \frac{1}{\varepsilon})$ ← useful later

(& can evaluate in poly time)

Why? given $\mathcal{A}$ learning $\mathcal{C}$

use $\mathcal{A}$ with $\varepsilon_0 = 1/4$

boost $\mathcal{A}$ to arbitrary ε

Boosting $\Rightarrow$

average case complexity insights:

Corr $\mathcal{C}$ learnable $\Rightarrow$ all concepts $c \in \mathcal{C}$ have poly sized ckts

Pf idea

$\forall\, c \in \mathcal{C}$, use $\varepsilon < \frac{1}{2^n}$ (e.g. no error)

in boosting

will output consistent hypothesis of poly size in $n (= \log \frac{1}{\varepsilon})$ & $|c|$

↖ # bits used by WL to describe c

that is poly time evaluatable.

$\Rightarrow$ poly sized ckt.

▨

Thm. Suppose f cannot be computed by poly sized ckts. Then there is a sequence of distributions $\{\mathcal{D}_n^*\}$ s.t. f is "average-case hard" on $\{\mathcal{D}_n^*\}_{n=1}^{\infty}$

- "average-case hard": no poly sized ckt gets f right more than $\frac{1}{2} + \frac{1}{\text{poly}(n)}$ of time
- need hard dist for each input size to make it well defined

Why nontrivial? output for any single input can be hardwired into ckt. $\Rightarrow$ no "hard" inputs. how to find a set of "collectively hard" inputs?

Pf idea

if not, f can be weakly-learned by poly sized ckts

(boosting) $\Rightarrow$ can strongly learn in size poly$(\log \frac{1}{\varepsilon}, \ldots)$

(previous corollary) $\Rightarrow$ can learn f with 0 error

$\Rightarrow$ f computable by poly-sized ckts