Lecture 22

distribution-free weak learning
$\Rightarrow$ strong learning } "boosting"

average vs. worst case complexity

Weak vs. Strong Learning

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

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

$\mathcal{A}$ outputs h s.t. $\Pr_{\mathcal{D}}[h(x) = c(x)] \geq \underbrace{\frac{1}{2}}_{\text{not good compared to } 1-\varepsilon \text{ or } 99\%} + \frac{\gamma}{2}$

↑ advantage over guessing

It was first conjectured that weak learning is easier than strong (i.e. ∃ fctns that can weakly learn but not strongly learn)

Surprise!!

Can "boost" a weak learner

__Thm__ if $\mathcal{C}$ can be weakly learned on __any__ dist $\mathcal{D}$ then $\mathcal{C}$ can be $\underbrace{\text{(strongly)}}_{\text{i.e. } \forall \varepsilon}$ learned

dependence on γ ?

δ ?

ε ?

Applications:

1) "theoretical"

- uniform distribution algorithms for
  poly term DNF
  weight-w poly threshold fctns

  low degree alg doesn't work well

  (Boosting + KM)

- Ave case vs. worst case complexity

2) practical: "Boosting"

Freund-Schapire

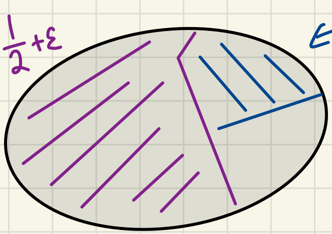
Good & Bad Ideas

1) simulate weak learner several times on same distribution & take

majority answer
<u>or</u>
best answer

- gives better confidence
- but doesn't reduce error - what if always get same answer?

2) filter out examples on which current hypothesis does well & run weak learner on part where you do badly



← $\frac{1}{2}+\varepsilon$ of non-purple

<u>Problem</u>: given <u>new</u> example, how do you know which section it is in?

3) Keep some samples on which you are ok in your filtering.
Always use majority vote on previous hypotheses to predict value of new samples.

history: Schapire, Freund-Schapire, Impagliazzo-Servedio-Klivans

Filtering Procedures:

- decide which samples to keep vs. throw out
- samples on which you guess
  - correctly: needed for checking future hypotheses
  - incorrectly: needed for improvement

The setting

- Given labelled examples

$$(x_1, f(x_1)) \quad (x_2, f(x_2)) \ldots$$

$$x_i \in_R \mathcal{D}$$

$$f \in \mathcal{C} \qquad \text{"target function"}$$

- Given weak learning alg WL which weakly learns $\Big(\underbrace{\text{advantage } \tfrac{\gamma}{2}}_{\substack{\| \\ \text{error } \underbrace{\frac{1}{2} - \frac{\gamma}{2}}_{= \beta}}}\Big)$ on $\underline{\text{any}}$ $\underbrace{\text{dist } \mathcal{D}}$

$$\text{error}_{\mathcal{D}}(h) = \Pr_{x \in \mathcal{D}}[f(x) \neq h(x)]$$

Plan:

1. simple "modest" accuracy boosting procedure

2. recursively use ① to drive down error

Part I: Modest Improvement

Given: oracle to f

example oracle $\mathcal{D}$

weak learning algorithm WL

Algorithm:

note: now also have oracle for h_1!

$h_1 \leftarrow$ run WL on $\mathcal{D}$ for fctn f

Create example oracle $\mathcal{D}_2$:

Question - how many samples of $\mathcal{D}$ needed per output sample of $\mathcal{D}_2$?

flip coin:

"normalize" $\mathcal{D}$ to make h_1 err half the time

so $err_{\mathcal{D}_2}(h_1) = \frac{1}{2}$

heads - draw examples from $\mathcal{D}$ until find x s.t. $h_1(x) = f(x)$ "h_1 correct"
output x

tails - draw examples from $\mathcal{D}$ until find x s.t. $h_1(x) \neq f(x)$ "h_1 incorrect"
output x

(Algorithm cont.)

note $err(h_2) < \frac{1}{2}$ $\mathcal{D}_2$ so $h_1 \neq h_2$

$h_2 \leftarrow$ run WL on $\mathcal{D}_2$ for f

Create example oracle $\mathcal{D}_3$:

draw examples from $\mathcal{D}$ until find x s.t. $h_1(x) \neq h_2(x)$

output x

$h_3 \leftarrow$ run WL on $\mathcal{D}_3$ for f

output $h \equiv maj(\underbrace{h_1, h_2, h_3}_{\text{on } x \text{, evaluate } h_1(x), h_2(x), h_3(x) \text{ \& output majority answer}})$

Error Analysis of "Modest Improvement"

define 3 error probabilities:

$$\beta_1 = \Pr_{\mathcal{D}}[h_1(x) \neq f(x)]$$

$$\beta_2 = \Pr_{\mathcal{D}_2}[h_2(x) \neq f(x)]$$

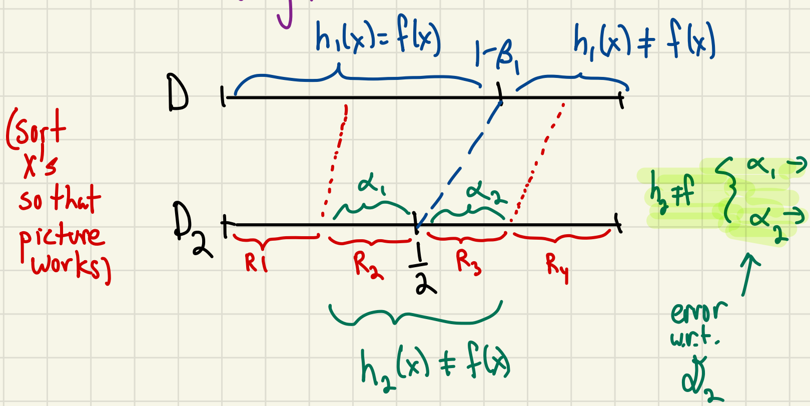
$$\beta_3 = \Pr_{\mathcal{D}_3}[h_3(x) \neq f(x)]$$

Observation:

if $h_1(x) = f(x)$ then $D(x) = 2(1-\beta_1) D_2(x)$

" " $\neq$ " " $D(x) = 2\beta_1 D_2(x)$

Why?



4 regions:

R_1:	$h_1 = h_2 = f$	majority correct
R_2:	$h_1 = f \neq h_2$	h_3 helps?
R_3:	$h_1 = h_2 \neq f$	majority incorrect
R_4:	$h_1 \neq h_2 = f$	h_3 helps?

h_3 "good" in R_2, R_4

On reweighting between D + D_2: (proof of observation)

for x s.t. $h_1(x) = f(x)$:

Total wt of x s.t. $h_1(x) = f(x)$

goes from $1-\beta_1$ to $1/2$

+ relative wts of x's stays same

$$\sum_{\substack{x \text{ s.t.} \\ h_1(x) = f(x)}} D(x) = 1 - \beta_1$$

$$\sum_{\substack{x \text{ s.t.} \\ h_1(x) = f(x)}} \underbrace{D(x) \cdot \alpha}_{D_2(x)} = \frac{1}{2}$$

$$1 - \beta_1 = \frac{1}{2 \cdot \alpha}$$

So, $D_2(x) = D(x) \cdot \alpha$

$$= \frac{1}{2 \cdot (1-\beta_1)} \cdot D(x)$$

+ $D(x) = 2(1-\beta_1) D_2(x)$

for x s.t. $h_1(x) \neq f(x)$:

$$\beta_1 = \frac{1}{2\alpha'}$$

so $D_2(x) = \alpha' \cdot D(x) = \frac{1}{2 \cdot \beta_1} D(x)$

+ $D(x) = 2\beta_1 D_2(x)$

More general observation:

$$\boxed{\begin{aligned}\forall S,\ \Pr_{x\in D}[x\in S] = {} & 2(1-\beta_1)\cdot \Pr_{x\in D_2}[h_1(x)=f(x) \wedge x\in S] \\ & + 2\beta_1 \cdot \Pr_{x\in D_2}[h_1(x)\neq f(x) \wedge x\in S] \quad (*)\end{aligned}}$$

Bounding error of WL:

$\beta \leftarrow$ error guarantee (by assumption) on WL's output

$g(\beta) \leftarrow 3\beta^2 - 2\beta^3$

<u>Main Lemma</u>: $err_D(h) \le g(\beta)$

note: $g(\beta) \le \beta$ but how much better?

not always better since $g(\frac{1}{2}) = \frac{1}{2}$

Proof $err_D(h)$ from 2 types:

Type ① x s.t. $h_1(x) = h_2(x) \neq f(x)$ (both h_1 + h_2 wrong)

"lost cause" case

so no matter whether $h_3(x)$ wrong or right $h = maj(h_1, h_2, h_3)$ will be wrong on x

Type ② x s.t. $h_1(x) \neq h_2(x)$

here $h_3(x)$ determines if h correct

defined on next page

$$\text{so } err_D(h) = \underbrace{\Pr_{x \in D}[h_1(x) \neq f(x) + h_2(x) \neq f(x)]}_{2\beta_1^{\alpha}2}$$

$$+ \underbrace{\Pr_{x \in D}[h_3(x) \neq f(x) \mid h_1(x) \neq h_2(x)]}_{\text{def of } \beta_3} \cdot \Pr_{x \in D}[h_1(x) \neq h_2(x)]$$

$$\leq \Pr_{x \in D}[h_1(x) \neq f(x) + h_2(x) \neq f(x)]$$

$$+ \underbrace{\beta}_{\beta_3 \leq \beta} \cdot \Pr_{x \in D}[h_1(x) \neq h_2(x)]$$

equation (0)

Type ② — Calculating $\Pr_{\mathcal{D}}[h_1(x) \neq h_2(x)]$:

Partition $\mathcal{D}_2$ into 2 parts:

1) x s.t. $h_1(x) = f(x)$ — regions R_1, R_2
2) " " $h_1(x) \neq f(x)$ — regions R_3, R_4

R_2: $\alpha_1 =$ err of h_2 wrt $\mathcal{D}_2$ on part 1 $= \Pr_{x \in \mathcal{D}_2}[h_1(x) = f(x) \;\&\; h_2(x) \neq f(x)]$

R_3: $\alpha_2 =$ " " " " " " " 2 $= \Pr_{x \in \mathcal{D}_2}[h_1(x) \neq f(x) \;\&\; h_2(x) \neq f(x)]$

$$\alpha_1 + \alpha_2 = \beta_2$$

R_2:

Then $\Pr_{x \in \mathcal{D}}[h_1(x) = f(x) \wedge h_2(x) \neq f(x)]$

use reweighting calculations from before

$$= 2 \cdot (1-\beta_1) \Pr_{x \in \mathcal{D}_2} \Big[\underbrace{h_1(x) = f(x) \wedge h_2(x) \neq f(x)}_{\alpha_1}\Big]$$

$$= 2(1-\beta_1)\alpha_1$$

need to reweight these two cases differently

And $\Pr_{x\in\mathcal{D}_2}[\underbrace{h_1(x)\neq f(x)}_{\substack{\Pr=\frac{1}{2} \\ \text{by construction} \\ \text{of } \mathcal{D}_2}} \wedge h_2(x)=f(x)] = \frac{1}{2}-\alpha_2$ ← defn of α_2

so $\Pr_{x\in\mathcal{D}}[h_1(x)\neq f(x) \wedge h_2(x)=f(x)] = 2\beta_1\left(\frac{1}{2}-\alpha_2\right)$

Putting together:

$$\Pr_{x\in\mathcal{D}}[h_1(x)\neq h_2(x)] =$$

$$\Pr_{x\in\mathcal{D}}[h_1(x)=f(x) \wedge h_2(x)\neq f(x)]$$

$$+ \Pr_{x\in\mathcal{D}}[h_1(x)\neq f(x) \wedge h_2(x)=f(x)]$$

$$= 2(1-\beta_1)\alpha_1 + 2\beta_1\left(\frac{1}{2}-\alpha_2\right)$$

Finally:

$$\text{err}_{\mathcal{D}}(h) \leq 2\beta_1\alpha_2 + \beta\left(2(1-\beta_1)\alpha_1 + 2\beta_1\left(\tfrac{1}{2}-\alpha_2\right)\right)$$

$$\vdots$$

assume $\beta=\beta_1=\beta_2$ & use $\alpha_1+\alpha_2=\frac{\beta}{2}$

$$\leq \beta^2 + 2\beta(1-\beta)(\alpha_1+\alpha_2) \leq 3\beta^2 - 2\beta^3$$

∎

Part II Recursive accuracy boosting

one application takes error $\beta \to \leq 3\beta^2 - 2\beta^3$

we want tiny error

main idea: Recursion

Algorithm: given ρ, D'

if $\rho \geq$ promised error of WL, return result of WL on D'

else:

$\beta \leftarrow g^{-1}(\rho)$ (error required from level below to get error $\leq \rho$ here)

define D_2' + D_3' as in "modest boost"

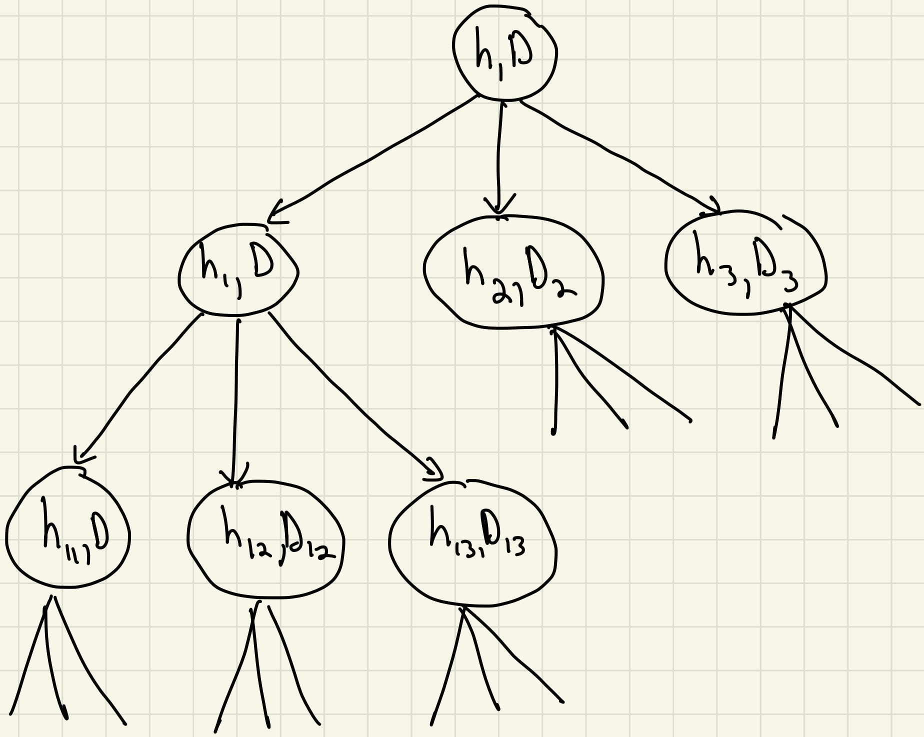
$h_1 \leftarrow$ strong learn $(\beta, Ex(f, D'))$

$h_2 \leftarrow$ strong learn $(\beta, Ex(f, D_2'))$

$h_3 \leftarrow$ strong learn $(\beta, Ex(f, D_3'))$

$h \leftarrow$ maj (h_1, h_2, h_3)

return h



issues:

- how many recursive calls?

  depth & size of recursion tree

- how many samples to construct filtered distributions?

samples:

problem ... filtering can take a while

but good news!

if it takes a while to find good samples,

then we've already learned well!

e.g. to find samples s.t. $h_1(x) = f(x)$

more than half should satisfy

(in any case if few such samples,

can output $\overline{h_1}$ to get good

approx to f)

to find samples s.t. $h_1(x) \neq f(x)$:

if can't find, then h_1 is good

approx to f

to find samples s.t. $h_1(x) \neq h_2(x)$:

if always agree, then don't need h_3

depth of recursion:

assume WL advantage γ is $\frac{1}{2} \Longrightarrow \beta \leq \frac{1}{4}$

(but also works if $\gamma = \Omega(\frac{1}{n^c})$)

Claim if $\beta \leq \frac{1}{4}$, $g(\beta) \leq 3\beta^2 = \frac{1}{3}(3\beta)^2$

error decrease in depth k is down to

$$\leq \frac{1}{3}(3\beta)^{2^k}$$

Important Consequences:

$\Longrightarrow$ $k = \theta(\log\log(\frac{1}{\varepsilon}))$ depth suffices to get error $\leq \varepsilon$

$\Longrightarrow$ size $3^{\log\log\frac{1}{\varepsilon}}$ $(\approx \theta(\log 1/\varepsilon))$

$\times\, s$ $\longleftarrow$ description of weak learning hypothesis

$= O(s \log 1/\varepsilon)$

suffices to describe circuit

$\mathcal{C}$ is concept class where s bound size of concepts

<u>Thm</u> $\mathcal{C}$ learnable $\Rightarrow \exists$ efficient algorithm

- using $\dfrac{\text{poly}(n, s, \log 1/\varepsilon, \log 1/\delta)}{\varepsilon}$ samples & time

- Outputs hypotheses of size

  $\text{poly}(n, s, \log \frac{1}{\varepsilon})$

  (& can evaluate in poly time)

<u>Why</u>? given $\mathcal{A}$ learning $\mathcal{C}$

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

boost $\mathcal{A}$ to arbitrary ε

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

Pf idea

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

will output consistent hypothesis of poly size in n & $|c|$ ← # bits to describe c

that is poly time evaluatable.

$\Rightarrow$ poly sized ckt.

$\blacksquare$

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

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

Pf idea

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

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

$\Rightarrow$ can learn f with 0 error

$\Rightarrow$ f computable by poly-sized ckts