Today:

Weak vs. Strong Learning

via "Boosting"

Def. Algorithm $\mathcal{A}$ weakly-"PAC learns" concept class $\mathcal{C}$ if $\forall\, c \in \mathcal{C}$

& $\forall$ dists $\mathcal{D}$

$\forall\ \cancel{\varepsilon}, \delta > 0$

with prob $\geq 1-\delta$

given labelled examples of $c \leftarrow (x, c(x))$ from dist $\mathcal{D}$

$\mathcal{A}$ outputs h s.t. $\Pr_{x \sim \mathcal{D}}[h(x) \neq c(x)] \leq \cancel{\varepsilon}\ \frac{1}{2} - \frac{\gamma}{2}$

γ is "advantage"

is Weak learning easier than strong-learning ?

regular PAC learning

surprisingly "no"!

<u>Thm</u> if $\mathcal{C}$ can be weakly learned on <u>any</u> dist $\mathcal{D}$ then $\mathcal{C}$ can be (strongly) $\underbrace{\text{PAC}}_{\forall \varepsilon}$ -learned

Applications:

1) "Theoretical"

- Boosting + KM → unif dist learning algs for poly term DNF (better than low deg alg)

- insights into average case & worst case complexity

2) Practical -

many boosting algs

Freund Schapire ... (many many)

history: Schapire

Freund, Schapire

⋮

lots more

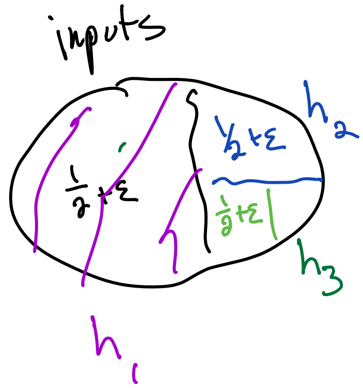
Good & Bad ideas

1) simulate weak learner many times on same distribution & take {majority answer / best answer}

$\Rightarrow$ better confidence (δ)

but doesnt reduce error

2) filter out examples on which current hypothesis does well & rerun weak learner on part where we did badly



training phase:
$x, f(x)$

later:
given x, predict $f(x)$
here we don't know if h_1, h_2 or h_3 correct
which section are we in?

3) Keep some "good" samples in filtering
use majority vote on all hypothesis $h_1 h_2 h_3$
(don't need to know "color" of x)

Filtering Procedures

- decide which samples to keep/throw
- need keep some samples on which you did well
  but weight more on those on which you need to improve

The setting:

- Given labelled examples $(x_1, f(x_1))$ $(x_2, f(x_2))$

  $x_i \in \mathcal{D}$

  target fctn $\rightarrow$ $f \in \mathcal{C}$

- Given weak learning alg WL which weakly learns $\left\{ \begin{array}{l} \text{advantage } \gamma \\ \text{error } \underbrace{\frac{1}{2} - \gamma/2}_{\beta} \end{array} \right\}$ on any dist $\mathcal{D}'$

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

The Plan:

- simple "modest" boosting procedure (error slightly improves)
- recursively use ↑ to drive down error

Part I: Modest improvement

Algorithm: Given oracle to f, $\mathcal{D}$ & WL

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

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

flip coin;

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

output x

Tails – " " " " " " " " $h_1(x) \neq f(x)$ "h_1 incorrect"

output x

normalize $\mathcal{D}$ s.t. errs half the time

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

how many samples?

H: ≤ 2

T: if $\geq \frac{1}{\varepsilon}$ then done!

$err_{\mathcal{D}_2}(h_2) < 1/2$ $\rightarrow$ $h_2 \leftarrow$ run WL on $\mathcal{D}_2$

so $h_1 \neq h_2$

create example oracle $\mathcal{D}_3$

draw examples until find x s.t. $h_1(x) \neq h_2(x)$

output x

if need too many samples $(> \frac{1}{\varepsilon})$ skip

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

output $h \equiv maj(h_1, h_2, h_3)$

ie. evaluate $h_1\ h_2$ & h_3 on x & output most common answer

Error analysis:

error $\leq \beta$ promised by WL

$$\left.\begin{aligned} \beta_1 &= \Pr_{\mathcal{D}}[h_1(x) \neq f(x)] \\ \beta_2 &= \Pr_{\mathcal{D}_2}[h_2(x) \neq f(x)] \\ \beta_3 &= \Pr_{\mathcal{D}_3}[h_3(x) = f(x)] \end{aligned}\right\}$$

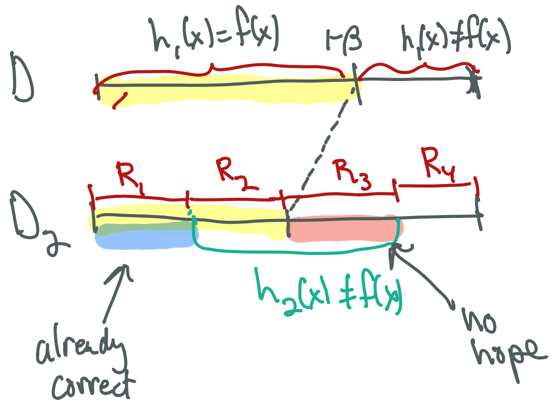
for simplicity (worst case) assume $\beta = \beta_1 = \beta_2 = \beta_3$

Observation

If $h_1(x) = f(x)$ then $\mathcal{D}(x) = 2(1-\beta_1)\,\mathcal{D}_2(x)$

$\neq$ $\qquad = 2\beta_1\,\mathcal{D}_2(x)$

why?



R_1: $h_1 = h_2 = f$ already maj correct

R_2: $h_1 = f \neq h_2$

R_3: $h_1 = h_2 \neq f$ already maj incorrect

win? R_4: $h_1 \neq h_2 = f$

$h_3 \geq \frac{1}{2} + \frac{\delta}{2}$ good in $R_2 + R_4$

total wt of x st. $h_1(x)=f(x)$ goes from

$1-\beta$ to $1/2$

& rel wts of these x's stay same

$$\left.\begin{array}{l}\displaystyle\sum_{\substack{x \text{ st} \\ h_1(x)=f(x)}} D(x) = 1-\beta \\ \displaystyle\sum_{\substack{x \text{ st} \\ h_1(x)=f(x)}} \left(D(x)\cdot\alpha\right) = \frac{1}{2}\end{array}\right\} \quad 1-\beta = \frac{1}{2\alpha}$$

so $\quad D_2(x) = D(x)\cdot\alpha = \dfrac{1}{2(1-\beta)} D(x)$

$\quad$ & $D(x) = 2(1-\beta) D_2(x)$

β = err ~~of output~~ of WL guarantee

Main Lemma

$$\mathrm{err}_{D}(h) \le \underbrace{3\beta^2 - 2\beta^3}_{\equiv g(\beta)}$$

$$g(\beta) \ll \beta$$

idea of pf:

$err_D(h)$ has 2 types:

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

so h_3 can't fix

2) x s.t. $h_1(x) \neq h_2(x)$

here h_3 decides if h correct

$$(0) \quad err_D(h) = \Pr_{x \in D}[h_1(x) \neq f(x) \text{ \& } h_2(x) \neq f(x)]$$

$$+ \underbrace{\Pr_{x \in D}[h_3(x) \neq f(x) \mid h_1(x) \neq h_2(x)]}_{\text{def of } \beta_3 = \text{error of WL on } D_3} \cdot \Pr_{x \in D}[h_1(x) \neq h_2(x)]$$

$\alpha_1 = \Pr_{x \in D_2}[h_1(x) = f(x) \text{ \& } h_2(x) \neq f(x)]$ region 2

$\alpha_2 = \Pr_{x \in D_2}[h_1(x) \neq f(x) \text{ \& } h_2(x) \neq f(x)]$ region 3

$\alpha_1 + \alpha_2 = \beta_2$

Then

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

$$= 2 \cdot (1-\beta_1) \underbrace{\Pr_{x \in D_2}[h_1(x) = f(x) \wedge h_2(x) \neq f(x)]}_{\alpha_1}$$

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

$$\Pr_{x \in D_2}[h_1(x) \neq f(x) \wedge h_2(x) = f(x)] = \tfrac{1}{2} - \alpha_2$$

So $\Pr_{x \in D}[h_1(x) \neq f(x) \wedge h_2(x) = f(x)] = 2\beta_1(\tfrac{1}{2} - \alpha_2)$ (1)

Putting together:

$$\Pr_{x \in D}[h_1(x) \neq h_2(x)] = 2(1-\beta_1)\alpha_1 + 2\beta_1(\tfrac{1}{2} - \alpha_2)$$

Also $\Pr_{x \in D}[\text{both } h_1 \& h_2 \text{ wrong}] = \underbrace{2\beta_1}_{\substack{\text{fix} \\ D_2 \to D_1}} \underbrace{\alpha_2}_{\text{def}}$ (2)

(1) + (2) put into previous

$$err_D(h) \leq 2\beta_1\alpha_2 + \beta[2(1-\beta)\alpha_1 + 2\beta(\tfrac{1}{2} - \alpha_2)$$

$\vdots$

$$\leq 3\beta^2 - 2\beta^3 \quad \blacksquare$$

$$g(y) = 3y^2 - 2y^3$$

Part II Recursive Accuracy Boosting

Strong learning Algorithm:

given ρ, D'

if ρ can be achieved directly from WL then just call WL & return result

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

{ get D_2' & D_3' as in "modest boost" }

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

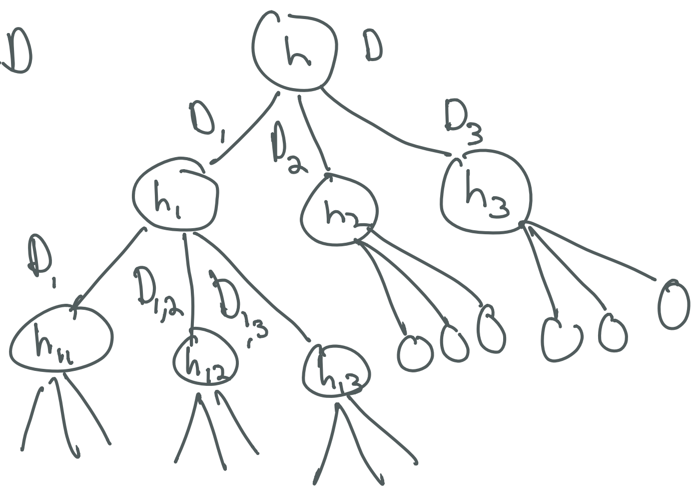
$h_2 \leftarrow$ strong learn $(\beta, Ex(f, D_2'))$

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

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

return h

$D_1 = D$



Sample Complexity:

how many recursive calls?

depth + size of recursion tree

how many samples to construct filters? ✓

depth of recursion:

if $\beta \le 1/4$ then $g(\beta) = 3\beta^2 - 2\beta^3$

$\le 3\beta^2$

in k steps $\le \frac{1}{3}(3\beta)^{2^k} \le \left(\frac{3}{4}\right)^{2^k}$

$\Rightarrow k = \Theta(\log\log\frac{1}{\varepsilon})$ depth suffices

size is $O(3^{\log\log 1/\varepsilon}) \approx \Theta(\log\frac{1}{\varepsilon})$

also does improve as long as $\beta \approx \frac{1}{2} - \frac{O(1)}{n^c}$ gives poly depth