Weak vs. Strong Learning

Def. Algorithm A weakly "PAC learns" concept class C

if $\forall\ c \in C$ & $\forall$ dists $\mathcal{D}$ $\exists\ \gamma > 0$

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

with prob $\geq 1-\delta$

given examples of c

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

↑ advantage

It was conjectured that distribution free weak learning was really weaker but surprise!

Can "boost" a weak learner

Thm if C can be weakly learned on any dist $\mathcal{D}$ then C can be (strongly) learned. ⇐ ie. $\forall \mathcal{D}_0$

Will prove for case of $\mathcal{D}_0 = \mathcal{U}$

Applications

1) "theoretical"
- Unif dist Algorithms for poly term DNF
  weight w - poly threshhold fctns } low degree alg doesn't work well
  ∴ (Boosting + KM)
- Ave case vs. worst case complexity

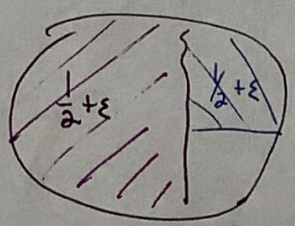
2) practical - Boosting
   Freund-Schapire

Good & Bad Ideas

1) simulate weak learner several times on same distribution & take majority answer -or- 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.



Problem: given a new example, how do you know which section it is in?

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

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

Filtering Procedures

- decide which samples to keep, which to throw out
- samples on which so far you guess correctly ← need for checking future hypotheses
  incorrectly ← need to improve on these

The setting

- Given labelled examples
  $(x_1, f(x_1)), (x_2, f(x_2)), \ldots$
  $x_i \in_R \mathcal{D}$
  target function → $f \in C$
- Given weak learning alg WL which weakly learns (advantage $\frac{\gamma}{2}$) on any dist $\mathcal{D}$
  or equivalently error $\frac{1}{2} - \frac{\gamma}{2} = \beta$
  (use $\text{error}_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[f(x) \neq h(x)]$)

Plan:

1. simple "modest" accuracy boosting procedure (error slightly improves)
2. recursively use ① to drive down error

Part I. Modest Improvement:

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

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

now also have oracle for h_1

Create example oracle $\mathcal{D}_2$: how many calls to $\mathcal{D}$ per output sample of $\mathcal{D}_2$?

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

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"

x s.t.

output x

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

note $err_{\mathcal{D}_2}(h_1) < 1/2$ → $h_2 \leftarrow$ run WL on $\mathcal{D}_2$ for f

so $h_2 \neq h_1$

Create example oracle $\mathcal{D}_3$: how many calls to $\mathcal{D}$ per sample of $\mathcal{D}_3$?

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

(ie. h is a concept which evaluates h_1, h_2, h_3 & outputs majority answer)

Error Analysis of "Midest Improvement": (3 error probabilities)

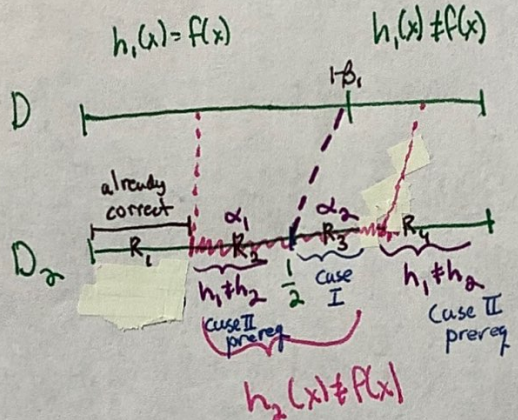
let $\beta_1 = \Pr_D[h_1(x) \neq f(x)]$ & $\beta_3 = \Pr_{D_3}[h_3(x) \neq f(x)]$

$\beta_2 = \Pr_{D_2}[h_2(x) \neq f(x)]$

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

" $h_1(x) \neq f(x)$ " $D(x) = 2\beta_1 D_2(x)$

why?



sort X's s.t. picture works

R_1: $h_1 = h_2 = f$ maj correct

$\leq \frac{1}{2} - \frac{\gamma}{2}$ { R_2: $h_1 = f \neq h_2$ h_3 helps?

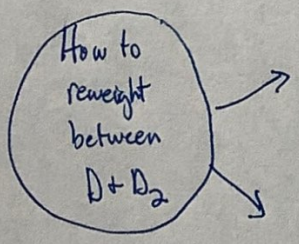
R_3: $h_1 = h_2 \neq f$ maj incorrect

R_4: $h_1 \neq h_2 = f$ h_3 helps?

h_3 is "good" on $R_2 + R_4$

total wt of x s.t. $h_1(x) = f(x)$ goes from $1-\beta_1$ to $\frac{1}{2}$

& relative wts of such x's stay same



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

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

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

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

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

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

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

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

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

More general observation:

$$\Pr_{x\in D}[x \in S] = 2(1-\beta_1) \Pr_{x\in D_2}[h_1(x) = f(x) \wedge x \in S] + 2\beta_1 \Pr_{x\in D_2}[h_1(x) \neq f(x) \wedge x \in S] \quad (*)$$

Let β be error guaranteed by assumption on output of WL
Let $g(\beta) = 3\beta^2 - 2\beta^3$

Main Lemma

$err_D(h) \leq g(\beta)$

note: $g(\beta) < \beta$ but by how much?
$g(1/2) = 1/2$

Pf.

$err_D(h)$ comes from 2 types:

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

so no matter if $h_3(x)$ is wrong or right, h (which is majority of h_1, h_2, h_3) will be wrong

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

so $h_3(x)$ decides if h is correct

$$\text{so } err_D(h) = \Pr_{x \in D}[h_1(x) \neq f(x) \text{ \& } h_2(x) \neq f(x)]$$
$$+ \Pr_{x \in D}[h_3(x) \neq f(x) \mid h_1(x) \neq h_2(x)] \cdot \Pr_{x \in D}[h_1(x) \neq h_2(x)]$$
$$= \Pr_{x \in D}[h_1(x) \neq f(x) \text{ \& } h_2(x) \neq f(x)]$$
$$+ \underbrace{\beta_3}_{\text{def of } \beta_3} \cdot \Pr_{x \in D}[h_1(x) \neq h_2(x)]$$
$$\leq \Pr_{x \in D}[h_1(x) \neq f(x) \text{ \& } h_2(x) \neq f(x)] + \beta \cdot \Pr_{x \in D}[h_1(x) \neq h_2(x)]$$

(0)

Finish 2nd term:

partition D_2 into 2 equal parts:

1) x s.t. $h_1(x) = f(x)$

2) " " " $\neq$ "

let α_1 = "err of h_2 (wrt D_2) on part 1" $\equiv \Pr_{x \in D_2}[h_1(x) = f(x) \wedge h_2(x) \neq f(x)]$

α_2 = " " " 2" $\equiv \Pr_{x \in D_2}[h_1(x) \neq f(x) \wedge h_2(x) \neq f(x)]$

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

"$x \in S$" $\equiv$ "$h_1(x) = f(x)$ $\wedge$ $h_2(x) \neq f(x)$"

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

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

$= \alpha_1$

$+ 2\beta_1 \cdot \Pr_{x \in D_2}[h_1(x) \neq f(x) \wedge h_1(x) = f(x) \wedge h_2(x) \neq f(x)]$

$= 0$

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

And $\Pr_{x \in D_2}[h_1(x) \neq f(x) \wedge h_2(x) = f(x)] = \frac{1}{2} - \alpha_2$ ← algebra + structure of D_2

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

Putting together

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

$$= 2(1-\beta_1)\alpha_1 + 2\beta_1(\tfrac{1}{2} - \alpha_2) \qquad (1)$$

2nd term:

Also

$$\Pr_{x \in D}\left[h_1(x) \neq f(x) \text{ & } h_2(x) \neq f(x)\right] = 2\beta_1\alpha_2 \qquad (2)$$

so putting (1) & (2) into (0):

$$\begin{aligned} err_D(h) &\leq 2\beta_1\alpha_2 + \beta \cdot \left[2(1-\beta_1)\alpha_1 + 2\beta_1\left(\tfrac{1}{2}-\alpha_2\right)\right] \\ &\qquad \text{assume } \beta_1 = \beta \\ &= \beta^2(1-2\gamma_1) + 2\beta\gamma_2(1-\beta) + 2\gamma_1\beta \\ &\vdots \\ &\leq \beta^2 + 2\beta(1-\beta)(\gamma_1+\gamma_2) \qquad \text{assume } \beta_2 = \beta \\ &\leq 3\beta^2 - 2\beta^3 \end{aligned}$$

∎

Part II Recursive Accuracy Boosting

one application of procedure takes error $\beta \rightarrow \leq 3\beta^2 - 2\beta^3$

but we want error $\leq$ tiny

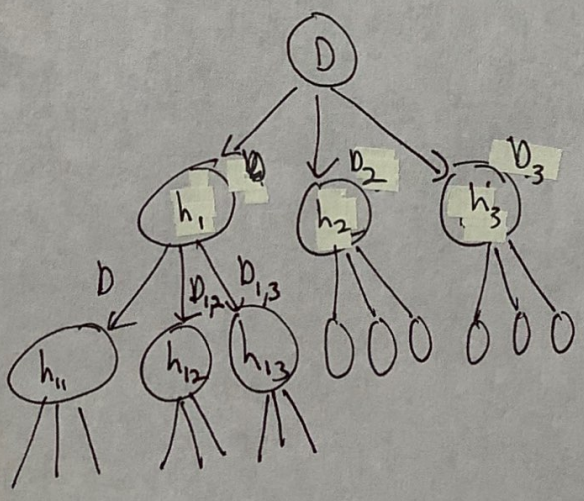
idea: recursion

Strong-learn Algorithm: (first try)

given ρ, D'
if $\rho \geq \frac{1}{2} - 1/p(\frac{1}{\epsilon})$ return result of WL on D'
$\beta \leftarrow g^{-1}(\rho)$ (error required from level below to get error $\leq \epsilon$ here)

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



Sample Complexity:

Main issues:
- how many recursive calls?
  depth + size of recursion tree
- how many samples required to construct filters?

Depth of recursion: (assume with advantage γ is $1/2$ $\Rightarrow$ error $\beta \le 1/4$
but also works for $\gamma = \Omega(\frac{1}{n^c})$
i.e. $\beta \to \frac{1}{2} - \frac{O(1)}{n^c}$

Claim: if $\beta \le 1/4$, $g(\beta) \le 3\beta^2 = \frac{1}{3}(3\beta)^2$
error decreases in k steps $\le \frac{1}{3}(3\beta)^{2^k} \le (3/4)^{2^k}$

$\Rightarrow$ $k = \Theta(\log\log(1/\varepsilon))$ depth suffices

$\Rightarrow$ size $3^{\log\log 1/\varepsilon} \approx \Theta(\log(1/\varepsilon))$ suffices

samples?

Problem: filtering could take a while

good news!
if filtering takes a while then we've already learned.
if can't find samples s.t. $h_1(x) = f(x)$
output h_1 (this can't happen though)
" " " " $h_1(x) \neq f(x)$
can halt + output h_1 for final answer
" " " " $h_1(x) \neq h_2(x)$
don't calculate h_3