Last time: started unit on boosting confidence + accuracy

- setup, framework of boosting
- proof-of-concept: 3-stage boosting, to go from 40% error $\rightsquigarrow$ 35.2% error
  - described $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$, how to sim. each $EX(c, \mathcal{D}_i)$

Today:

- analysis of 3-stage boosting, justification of why it gives 35.2%-error h wrt $\mathcal{D}$
- discuss ext. to "full" booster ($1-\varepsilon$ accuracy)
- " boosting over a fixed sample
- AdaBoost: simple, practical booster over a fixed sample; start its analysis

Questions?

Our combined "final" h is $MAJ(h_1, h_2, h_3)$ ($\mathcal{D} = \mathcal{D}_1$)

To show: $\Pr_{x \sim \mathcal{D}}[h(x) \neq c(x)] = 35.2$

Divide X into 4 regions, acc. to h_1, h_2 + c:

R_1 : x s.t.	R_2 : x s.t.	R_3 : x s.t.	R_4 : x s.t.
$h_1(x) = c(x)$, $h_2(x) = c(x)$	$h_1(x) = c(x)$, $h_2(x) \neq c(x)$	$h_1(x) \neq c(x)$, $h_2(x) \neq c(x)$	$h_1(x) \neq c(x)$, $h_2(x) = c(x)$.
$\mathcal{D}_2(R_1) = 0.5 - \rho$	$\mathcal{D}_2(R_2) = \rho$	$\mathcal{D}_2(R_3) = 0.4 - \rho$	$\mathcal{D}_2(R_4) = 0.1 + \rho$
$\mathcal{D}_1(R_1) = \frac{6}{5}(0.5 - \rho)$	$\mathcal{D}_1(R_2) = \frac{6}{5} \cdot \rho$	$\mathcal{D}_1(R_3) = \frac{4}{5}(0.4 - \rho)$	$\mathcal{D}_1(R_4) = \frac{4}{5}(0.1 + \rho)$

h_3 wrong 40% of time on $\mathcal{D}_1$, 100% of time on

Define $\rho := \mathcal{D}_2[R_2]$. What else can we figure out?

Recall $\mathcal{D}_2[x : h_2 \neq c(x)] = 0.4$ (WL guarantee), so $\mathcal{D}_2(R_3) = 0.4 - \rho$.

Recall also $\mathcal{D}_2[x : h_1(x) \neq c(x)] = 0.5$ (choice of $\mathcal{D}_2$),

so $\mathcal{D}_2(R_4) = 0.1 + \rho$ & $\mathcal{D}_2(R_1) = 0.5 - \rho$

Recalling how $\mathcal{D}_2$ was obt. from $\mathcal{D}_1$ by $5/6$, $5/4$ rescaling, can "undo" it & get $\mathcal{D}_1$'s wts of each R_i:

$\mathcal{D}_3$? Nonzero wt only on R_2, R_4; & on these, $\mathcal{D}_3$ is just a rescaling of $\mathcal{D}_1$.

h? Correct on all of R_1, incorrect on all of R_3; on $R_2 \cup R_4$, $h = c$ iff $h_3 = c$.

$\mathcal{D}_3, h_3$: $\Pr_{x \sim \mathcal{D}_3}[h_3(x) = c(x)] = 0.6$, i.e.

$\Pr_{x \sim \mathcal{D}_3}[h_3(x) \neq c(x)] = 0.4$.

So under $\mathcal{D} = \mathcal{D}_1$, h is wrong 40% of time on $R_2 \cup R_4$

So all in all,

$$\Pr_{\mathcal{D} = \mathcal{D}_1}[h(x) \neq c(x)] = 1 \cdot \mathcal{D}_1(R_3) + 0.4 \cdot \mathcal{D}_1(R_2 \cup R_4)$$

$$= \tfrac{4}{5}(0.4 - \rho) + 0.4\left(\tfrac{6}{5}\cdot\rho + \tfrac{4}{5}(0.1+\rho)\right)$$

$$= .352$$

$\tfrac{4}{5}(-\rho) + (0.4)\cdot\tfrac{6}{5}\rho + (0.4)\cdot\tfrac{4}{5}(0.1)\rho = 0$ (:o)

We went from 40% error to 35.2% error.
Can view 3-stage thing as new WL that gets 35.2% error; run 3-stage thing on this, & get acc < 35.2%

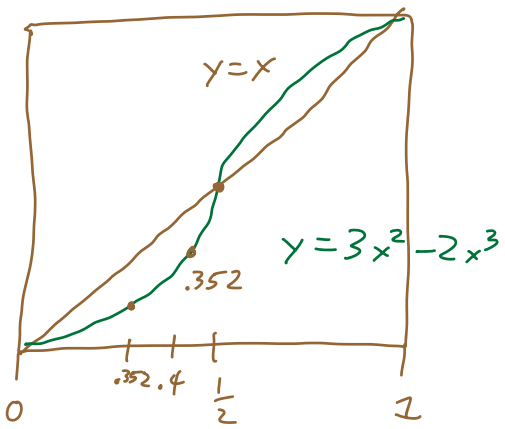
In general, if WL has error $\beta < \frac{1}{2}$, the hyp h coming from this 3-stage procedure will have error

$$3\beta^2 - 2\beta^3$$

can recurse!

If you apply $f(f(\ldots(0.4)\ldots))$

$O(\log\log\frac{1}{\varepsilon})$ iter. to get out $\leq \varepsilon$.



$f(x) = 3x^2 - 2x^3$

So overall booster, to get ε error, is a recursive tree of $O(\log\log\frac{1}{\varepsilon})$ levels of 3-step booster.

$3^{O(\log\log\frac{1}{\varepsilon})} =$ poly$(\log\frac{1}{\varepsilon})$ calls to WL overall.

Not practical. Next: much more practical booster

Boosting by Filtering (just did!)

vs

Boosting over a Fixed Sample

→ requires discarding a lot of data; :(

→ : • draw a fixed sample S of m data pts $(x_1, y_1) \ldots (x_m, y_m)$

$c(x_1)$ $\ll c(x_m)$

(ONLY DATA EVER USED)

- a dist $\longleftrightarrow$ a wt for each point.
  Initial $\mathcal{D} \leftrightarrow \mathcal{D}(x_i) = \frac{1}{m}$ for each i.

  New $\mathcal{D}'$: new asst of wts to pts, s.t.
  $\mathcal{D}'(x_i) \geq 0$, $\sum_{i=1}^{m} \mathcal{D}'(x_i) = 1$.
  $w_1, \ldots, w_m$

- WL guarantee: for any weighting

  $w_1, \ldots, w_m$ st $w_i \geq 0$, $\sum_{i=1}^{m} w_i = 1$, if

  run WL alg using this dist, get hyp h s.t.

$$\sum_{\substack{i \in [m]: \\ h(x_i) \neq y_i}} w_i \leq \frac{1}{2} - \gamma$$

- Desideratum of booster: final h has

$$\sum_{\substack{i \in [m]: \\ h(x_i) \neq y_i}} \frac{1}{m} \leq \varepsilon. \quad \text{(or 0)}$$

AdaBoost: simple, powerful, practical alg for boosting over fixed sample.

Key ideas:

⭐ • given h_t + D_t, construct D_{t+1} by adjusting weights "like D_2 did" in 3-stage thing:
make it s.t. h_t has error $\frac{1}{2}$ under D_{t+1}.

• At end, combine $h_1, \ldots, h_T$ by taking weighted MAJ vote: wt of h_t depends on its acc. under D_t

→ lets AdaBoost adapt to diff acc levels of different h_i's: take adv. of good ones.

Ex: $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$:

- $\varepsilon_t = 0.4$: $\alpha_t = 0.202$
- $\varepsilon_t = 0.01$: $\alpha_t = 2.29$

AdaBoost update rule analogous to WM alg:

WM		AdaBoost
expert i	$\longleftrightarrow$	i^{th} example for AdaBoost
pred. of expert i, trial t	$\longleftrightarrow$	$h_t(x_i)$
trial t	$\longleftrightarrow$	t^{th} run of WL
wt of expert i at trial t	$\longleftrightarrow$	$D_t(x_i)$

But now mistakes increase wt.

Next time: • justify ☆

• see that AdaBoost "puts more wt on pts that it just got wrong"

• state + prove thm on AdaBoost performance.