Lecture 20

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

(if time) average vs. worst case complexity

Last time: algorithm that weakly learns monotone fctns

↑ Slightly better than random guess

What good is weak learning?

unclear

here can only weakly learn on uniform distribution

ability to weakly learn on all distributions

$\Rightarrow$ ability to strongly learn

[Schapire]

"boosting"

Weak vs. Strong Learning

Def. Algorithm A "(strongly) PAC learns" concept class C

if $\forall c \in C$ + $\forall$ dists $\mathcal{D}$ $\forall \varepsilon, \delta$

given examples of c (labelled)

with prob $\geq 1-\delta$

A outputs h s.t. $\Pr_{\mathcal{D}}[h(x) = c(x)] \geq 1-\varepsilon$

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

It was first conjectured that weak learning is easier than strong (i.e. $\exists$ 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 (strongly) learned

i.e. $\forall \varepsilon$

dependence on γ ?
δ ?
ε ?

Applications:

1) "theoretical"

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

  (Boosting + KM)

low degree alg doesn't work well

- Ave case vs. worst case complexity

2) practical: "Boosting"

Freund-Schapire

Good & Bad Ideas

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

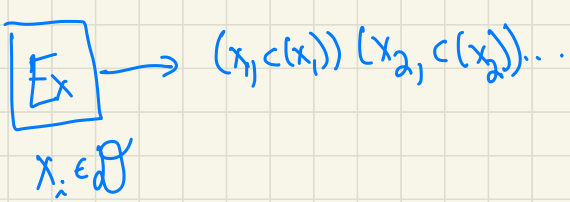
majority answer

<u>or</u>

best answer

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

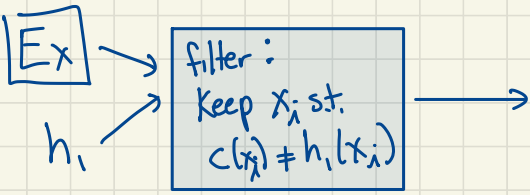
2) filter out examples on which current hypothesis does well & run weak learner on part where you do badly



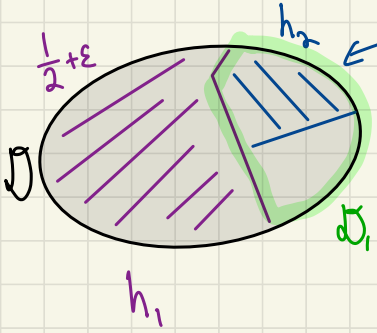
1. Run WL on Ex $\rightarrow h_1$

$\Pr_{x \in \mathcal{D}}[h_1(x) = c(x)] \geq \frac{1}{2} + \gamma$

2. use h_1 to "filter"



$\Pr_{x \in \mathcal{D}_1}[h_2(x) = c(x)] \geq \frac{1}{2} + \gamma$



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

e.g. if $h_1(x) = h_2(x) = b$ seems good to output b

but what if $h_1(x) \neq h_2(x)$?

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

... (lots and lots)

Filtering Procedures: (lots of these)

- 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

domain X

$\mathcal{C}$ concept class: $c \in \mathcal{C}$ maps $X \to \{0,1\}$

- Given labelled examples

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

$x_i \in_R \mathcal{D}$

$c \in \mathcal{C}$ "target function"

- Given weak learning alg WL which weakly learns (advantage γ) on any dist $\mathcal{D}$

(advantage γ) $=$ error $\frac{1}{2} - \gamma$

any dist $\mathcal{D}$: error$_{\mathcal{D}}(h) = \Pr_{x \in \mathcal{D}}[c(x) \neq h(x)]$

C is concept class where s bounds size of WL-description of concepts

(C ← parametrized by "size" n; s ← fctn of n)

Thm 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 A learning C

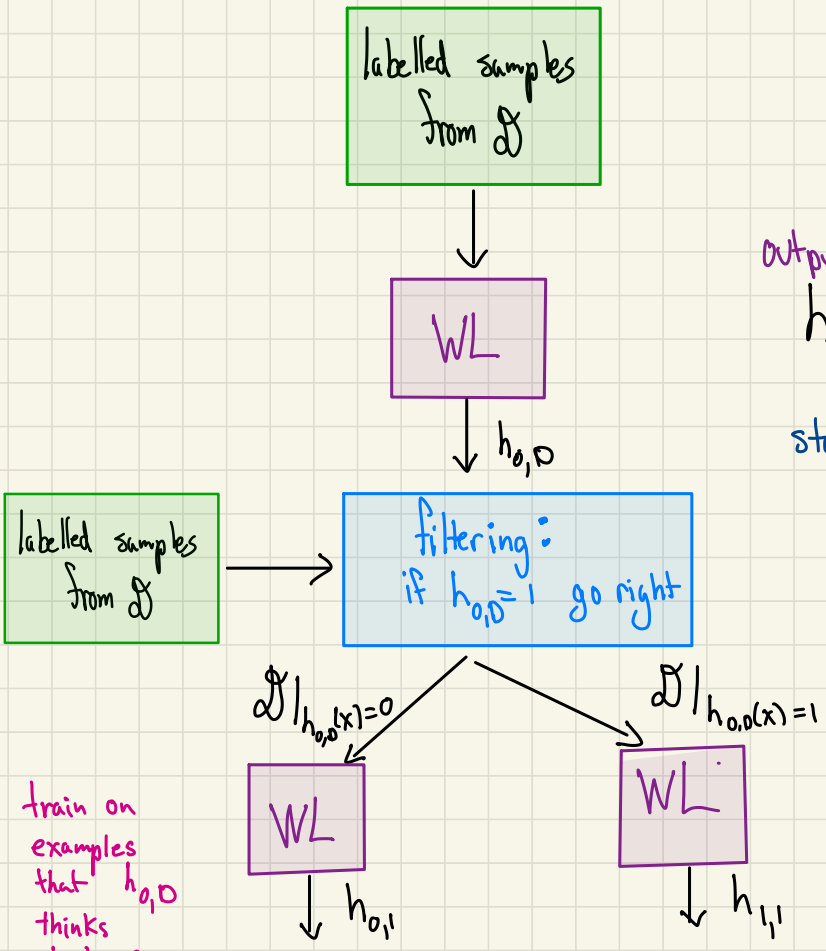
use A with $\varepsilon_0 = 1/4$

boost A to arbitrary ε

Idea: Filter by hypothesis value

One phase of filtering

assume $c: X \to \{0,1\}$



output of WL:

$h_{a,b}$

- stage
- trained on examples for which # labelled 1 is b

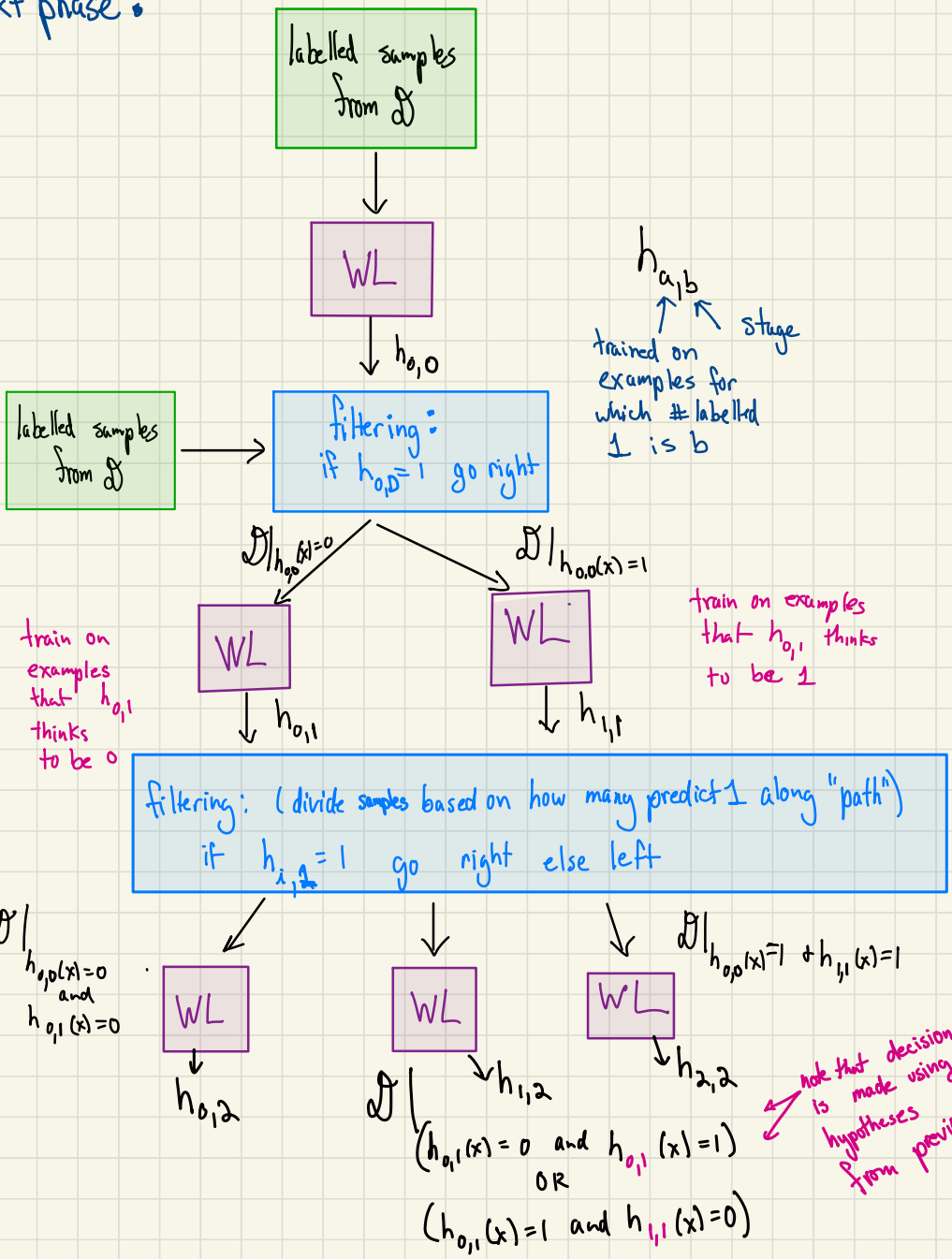
train on examples that $h_{0,0}$ thinks to be 0

train on examples that $h_{0,0}$ thinks to be 1

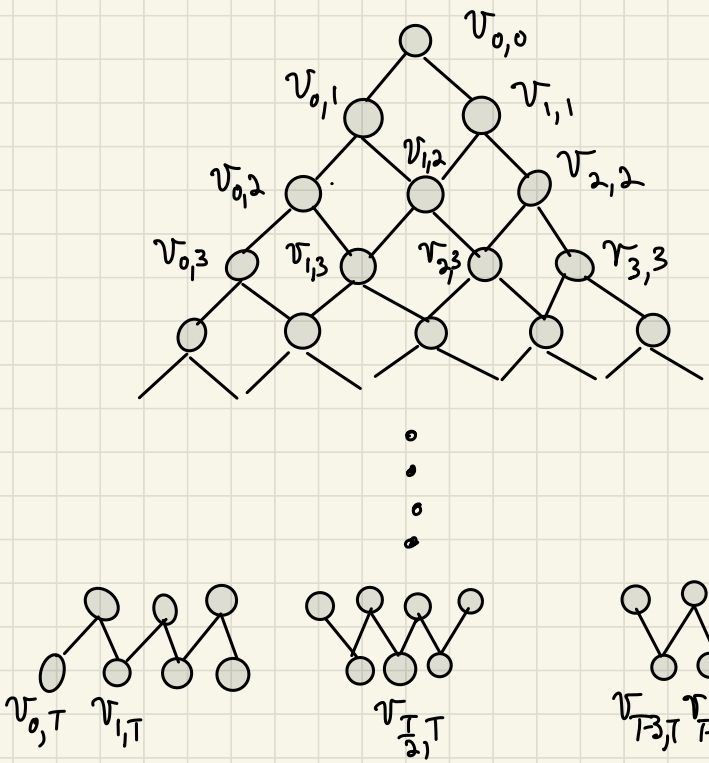
Idea: Filter by hypothesis value (cont.)

assume $c: X \to \{0,1\}$

next phase:



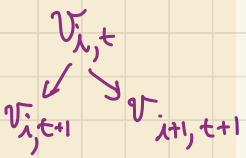
Putting together the filters:



T stages
$T+1$ layers

layer $0 \le t \le T$ has t nodes

each node in layer $\le T-1$ has 2 outgoing edges:



nodes in layer T have no outgoing edges

Node v_{ij} labelled by hypothesis h_{ij}

Routing of examples:

ignores labels of examples

given x, at node $v_{i,j}$

- evaluate $h_{i,j}(x)$ to get $y \in \{0,1\}$
- send on edge labelled by y, i.e. $v_{i+y, j+1}$

($i+y$: Left: $y=0$, Right: $y=1$; $j+1$: next level)

eventually reaches $v_{\ell,t}$

(ℓ: # of hypotheses evaluating to 1 on the way; t: level)

At t^{th} stage:

branching program induces $t+1$ different distributions

$$\mathcal{D}_{0,t}\ \mathcal{D}_{1,t}\ \mathcal{D}_{2,t} \cdots \mathcal{D}_{t,t}$$

where $\mathcal{D}_{j,t} = \mathcal{D}|_{\text{reach } v_{j,t}}$

run WL on each to get $h_{i,t}$ $\forall\, 0 \le i \le t$

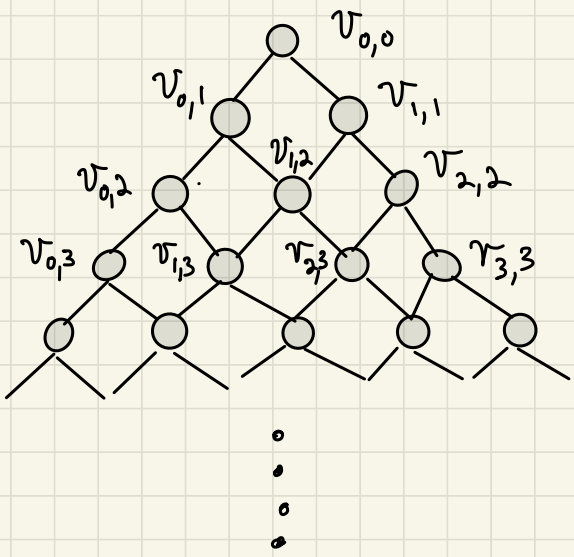
Final classifier:

given unlabeled x, route to level T

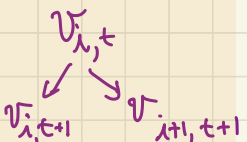
if reaches $v_{\ell,T}$ s.t. $\ell < T/2$ output 0

else output 1

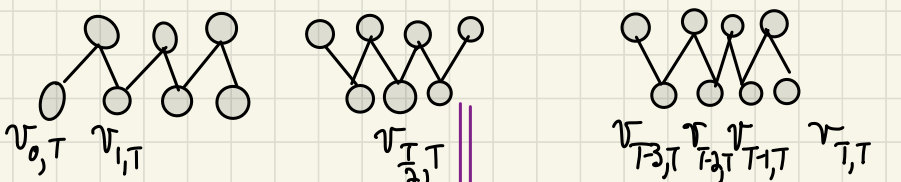
Putting together the filters:



T stages
$T+1$ layers
layer $0 \le t \le T$ has t nodes
each node in layer $\le T-1$ has 2 outgoing edges:



nodes in layer T have no outgoing edges



$\le T/2$

$> T/2$

"Majority":

for new unlabelled sample, if it ends up on this side output "0"

else, output "1"

Node v_{ij} labelled by hypothesis h_{ij}

Proof of correctness in special case:

Assume WL has 2-sided advantage wrt $\mathcal{D}$:

let $\mathcal{D}^+ \leftarrow \mathcal{D}|_{C(x)=1}$

$\mathcal{D}^- \leftarrow \mathcal{D}|_{C(x)=0}$

output of WL satisfies

(1) $\Pr_{x \in \mathcal{D}^+}[h(x)=1] \geq \frac{1}{2}+\gamma$

(2) $\Pr_{x \in \mathcal{D}^-}[h(x)=0] \geq \frac{1}{2}+\gamma$

note: all these should be labelled 1 →

these should be 0 →

(can show "reduction" to this case)

main idea:

show $x \in \mathcal{D}^+$ follows random walk biased right

$x \in \mathcal{D}^-$ " " " " left

Azuma's ≠ for sub-Martingales

(similar to Chernoff/Hoeffding when don't have complete independence)

def "submartingale" is sequence $Y_0 \cdots Y_T$ of r.v.'s with finite means & s.t. $\forall \ 1 \le i \le T$

$$E[Y_i | Y_0, Y_1, \ldots, Y_{i-1}] \ge Y_i$$

nuke: weaker than independence

Thm Azuma's ≠ for submartingales

Let $0 = Y_0, \ldots, Y_T$ be a submartingale with bounded differences

$$|Y_i - Y_{i-1}| \le c \quad \forall i$$

then

$$\Pr[Y_T \le -\lambda] \le e^{\left(-\lambda^2/2Tc^2\right)}$$

(will use $c=1$)

Thm let γ be $\in (0, \frac{1}{2})$

$\forall\ t = 0 .. T-1$, suppose each of $t+1$ calls to WL on $\mathcal{D}_{i,t}$ gives $h_{i,t}$ with 2-sided advantage γ w.r.t. $\mathcal{D}_{i,t}$

then final hypothesis has error $\leq e^{-\gamma^2 T}$

Pf.

bound error on $\mathcal{D}^+$ ($\mathcal{D}^-$ is identical)

$X_t \leftarrow$ [pick $x \in \mathcal{D}^+$; route to level t + reach $v_{i,t}$; output i]

Consider $X_t \mid X_0 \cdots X_{t-1}$

Condition on x reaching $v_{i,t-1}$

then $x \in (\mathcal{D}_{i,t-1})^+$

so $E[X_t \mid X_0 \cdots X_{t-1}] = X_{t-1} + \Pr_{x \in (\mathcal{D}_{i,t-1})^+}[h_{i,t-1}(x) = 1]$

$\geq X_{t-1} + \frac{1}{2} + \gamma$

since $h_{i,t-1}$ has 2-sided advantage

Define $Y_t \leftarrow X_t - t(\frac{1}{2}+\gamma)$ (γ: best known lower bound on $E[X_t]$) $\quad Y_0 = X_0$

note: conditioning on $Y_0 .. Y_t$ equivalent to " " $X_0 \cdots X_t$

so $E[Y_t | Y_0 .. Y_{t-1}] = E[X_t - t(\frac{1}{2}+\gamma) | Y_0 .. Y_{t-1}]$

$= E[X_t | Y_0 .. Y_{t-1}] - t(\frac{1}{2}+\gamma)$

$\geq X_{t-1} + \frac{1}{2} + \gamma - t(\frac{1}{2}+\gamma)$

$= X_{t-1} - (t-1)(\frac{1}{2}+\gamma) = Y_{t-1}$

$\Rightarrow$ sub-martingale

also $|Y_t - Y_{t-1}| = |X_t - X_{t-1} - (\frac{1}{2} + \gamma_{t-1})| \leq 1$

($X_t - X_{t-1}$: 0 or 1; $\frac{1}{2} + \gamma_{t-1} \in [\frac{1}{2}, 1]$)

So can use Azuma's $\neq$ for submartingales

which says $\Pr[Y_T \leq -\lambda] \leq e^{-\lambda^2/2T}$

taking $\lambda = \gamma \cdot T$ ✝ note

$$\Pr[Y_T \leq -\lambda]$$

$$= \Pr[X_T - T \cdot (\tfrac{1}{2} + \gamma) \leq -\gamma \cdot T]$$

$$= \Pr[X_T \leq \frac{T}{2}]$$

$$= \Pr_{x \in D^+}[h(x) = 0] \quad \text{for output hypothesis } h$$

↑ by defn of output hypothesis

So error rate $\leq e^{-(T\gamma)^2/2T}$

$$= e^{-T \cdot \gamma^2/2}$$

$\Rightarrow$ only need $T \sim \log 1/\varepsilon$

Sample / time complexity:

main issue: what if it takes a long time to get enough samples of either $\pm$ for one of the weak learners?

this will likely happen for t samples on $-$ side & viceversa →

(perhaps all the samples get filtered & placed in other buckets?)

Can "freeze" the node (won't contribute much to error)

freezing rule: if $v_{i,t}$ reached by "b"-labelled samples (either 0 or 1) with prob $\leq \frac{\varepsilon}{T(T+1)}$

then label $v_{i,t}$ by "$1-b$" & make it terminal ← other label

if not frozen, then get to $v_{i,t}$ with label b every $T(T+1)/\varepsilon$ samples!

Hypothesis: if no freezing on path, predict as majority of $h_{i,t}(x)$ on path (same as before)

else if freeze at level t node $v_{j,t}$

any example that reaches node $v_{j,t}$ gets labelled via $v_{j,t}$'s label

Why is the error ok?

- each frozen node contributes $\leq \frac{\varepsilon}{T(T+1)}$ to error
- $\leq T(T+1)$ frozen nodes $\Rightarrow \leq \varepsilon$ total error
- need to redo martingale analysis for unfrozen part (define intermediate process to make it similar)

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. Spose 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

$\Rightarrow$ 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