Lecture 18

More Boosting
Weak Learning

def. algorithm A weakly PAC learns concept class C if \( \exists J > 0 \) s.t.
\[
\forall c \in C \quad \forall \text{ dists } \mathcal{D},
\]
given examples of \( c \) according to \( \mathcal{D} \),
A outputs \( h \) s.t. \( \Pr_{\mathcal{D}}[h(x) \neq c(x)] < \frac{1}{2} - \frac{J}{2} \)

Then if \( C \) can be weakly PAC learned (on any \( \mathcal{D} \)), then
\( C^o \) can be (strongly) PAC learned.
**Weak vs. Strong Learning**

Def. Algorithm $A$ weakly "PAC learns" concept class $C$ if $\forall \mathcal{D} \subseteq C$ s.t. $\mathcal{D}$ dist $\mathcal{D} \quad \exists \delta > 0

\forall \delta, \delta' > 0 \quad (\delta' = \frac{\delta}{2} \text{ or } \frac{\delta}{2} \text{ doesn't affect})

with prob $\geq 1 - \delta$

given examples of $C$

$A$ outputs $h$ s.t. $\Pr[h(x) \neq c(x)] \leq \frac{\delta}{2} - \frac{\delta}{2}$

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

can "boost" a weak learner

Then if $C$ can be weakly learned on any dist $\mathcal{D}$ then $C$ can be (strongly) learned.
Applications

1) "Theoretical"
   - Unit dist Algorithms for poly kernel DNF weight w - poly threshold fnms
     (Boosting + KM)
   - Ave case vs. worst case

2) Practical - Boosting
   Freund-Schapire

Good & Bad Ideas

1) Simulate weak learner several times on
   same distribution & take majority answer
   - or -
   gives better confidence
   but doesn't reduce error, what if always get same answer?

2) Fill up 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
   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 \( \leftarrow \) need for checking
  future hypotheses
- incorrectly \( \leftarrow \) need to improve on

The setting

- Given labelled examples
  \( (x_1, f(x_1)), (x_2, f(x_2)), \ldots \)
  \( x_i \in \mathbb{R}^d \)
  \( f \in \mathbb{C} \)

- Given weak learning alg WL which weakly
  learns (advantage \( \frac{1}{2} \)) on any dist \( \mathcal{D} \)
Boosting Algorithm

Stage 0 (Initialize)

\[ D_0 \leftarrow 0 \]

run WL on \( D_0 \) to generate \((whp)\)

\[ C_i \quad \text{st:} \quad \Pr_{D_0} \left[ \hat{f}(x) = C_i(x) \right] \geq \tfrac{1}{2} + \tfrac{1}{2} \]

* For \( i = 1 \ldots T = O\left(\frac{1}{\epsilon^2}\right) \) stages, stage \( i \) \((\text{can stop if })\)

\[ \text{Majority}(C_1, \ldots, C_i) \text{ correct on } \geq 1 - \epsilon \text{ input} \)

(1) Construct \( D_i \) via "filtering procedure":

favor pts on which maj of \( C_1, \ldots, C_i \) don't do well

but also keep some other points \( \dddot{3} \)

Will specify soon

(2) run WL on examples from \( D_i \) to output

\[ C_{i+1} \quad \text{st:} \quad \Pr_{D_i} \left[ \hat{f}(x) = C_{i+1}(x) \right] \geq \tfrac{1}{2} + \tfrac{\epsilon}{2} \]

* output \( C = \text{MAJ}(C_1, \ldots, C_i) \)
Filtering procedure

Given new example $x, f(x)$ from example oracle

- if majority of $c_1, ..., c_n$ wrong, keep it
  \[ i.e. \quad i = \frac{n}{2} \]

- if large majority right, then discard
  \[ i.e. \quad \text{# right} - \text{# wrong} > \frac{1}{\gamma \varepsilon} \]
  \[ \text{or} \quad \text{# wrong} \leq \frac{i}{2} - \frac{1}{2 \gamma \varepsilon} \]

- else \quad \# right - \# wrong = \frac{\alpha}{\gamma \varepsilon} \quad \text{for} \quad 0 < \alpha \leq 1

  \[ \#\text{wrong} - \#\text{right} = \frac{-\alpha}{\gamma \varepsilon} \]

  So keep with prob = $1 - \alpha$

\[ \text{Prob of keeping} \quad 1 - \alpha \]

\[ \text{slope} = 2\gamma \varepsilon \]

\[ \frac{i}{2} - \frac{1}{2 \gamma \varepsilon} \]
Need to show:

1) Output is has nontrivial agreement with $f$.

2) # samples needed not too bad.

Why could it be bad?

- If throw out lots of samples, might need to wait a long time before WL can give an output.
- If throw out too many samples then you already have a good hypothesis.

$\uparrow$

Will stop if $\text{Maj}(C_1\ldots C_x)$ correct on $\geq 1-\varepsilon$ fraction of inputs.

- $\text{Maj}(C_1\ldots C_x)$ incorrect on $\geq \varepsilon$ fraction.

So filtering procedure outputs sample with prob $\geq \varepsilon$.

(If, in expectation, every $\varepsilon$ samples of off at least one makes $\text{Maj}$ throw the filtering system.)

$\Rightarrow$ filtering slows down sample collection by $\leq O(1/\varepsilon)$.

So let's focus on (1).
Notation

\[ R_c(x) = \begin{cases} 1 & \text{if } f(x) = c(x) \\ -1 & \text{if } f(x) \neq c(x) \end{cases} \]

\[ N_i(x) = \sum_{1 \leq j \leq i} R_{c_j}(x) \]

\[ M_i(x) = \begin{cases} 1 & \text{if } N_i(x) \leq 0 \\ 0 & \text{if } N_i(x) \geq \frac{1}{\varepsilon} \\ 1 - \varepsilon \cdot x \cdot N_i(x) & \text{o.w.} \end{cases} \]

Note that the new distribution on samples is proportional to \( M_i \):

\[ D_{M_i}(x) = \frac{M_i(x)}{\sum_{x} M_i(x)} \]

The distribution induced by \( M \) is

\[ D_{M}(x) = D_{M_i} \]

How correct are we with \( D_{M_i} \)?

\[ \text{Adv}_c(M) = \sum_{x} R_c(x) \cdot M_i(x) \]

\[ \Pr_{x \in D_{M_i}}[c(x) = f(x)] = \frac{1}{2} + \frac{\text{Adv}_c(M_i)}{2 \cdot \sum_{x} M_i(x)} \]

"Advantage" of \( c \) on \( M \)

\[ \approx \Pr[\text{correct}] - \Pr[\text{incorrect}] \]

\[ = 2 \cdot \Pr[\text{correct}] - 1 \]
Note:

If \( \sum M_i(x) \geq \varepsilon 2^n \)

\[ \text{Adv}_c(M_n) \equiv \gamma \cdot \varepsilon \cdot 2^n \]

**Begin Proof**

For input \( x \)

\[ A_x(x) \leq \sum_{0 \leq j \leq i-1} R_{c_{ij}}(x) M_j(x) \]

Claim \( A_x(x) \leq \frac{1}{\varepsilon^2} + \frac{3 \varepsilon}{2} \cdot i \)

- bounds advantage per input
- only helps after \( \frac{1}{\varepsilon^2} \) rounds

Plan for use of claim:

Consider large matrix:

\[(h,i)^{th} \text{ Entry: } R_{c_{ij}}(x) M_j(x) \]

\( x \)'s row sum = \( \sum_{0 \leq j \leq i} R_{c_{ij}}(x) M_j(x) = A_x(x) \)

\( i \)'th col sum = \( \sum_{x} R_{c_{ij}}(x) M_j(x) \)

\[ = \text{Adv}_{c_{ij}}(M_j) \leq \varepsilon \cdot \text{Adv}_c(M_j) \]

else algorithm stops
Goal: lower/upper bound average entry in matrix

lower bound:

- lower bound average entry in column via correctness of WL
- fact that algorithm proceeds

\[
\Rightarrow \text{lots of error} \\
\Rightarrow \sum_{x} M_{j}(x) \text{ is big} \\
\Rightarrow \text{lots of progress in WL in absolute terms}
\]

upper bound:

- upper bound rows via claim
- if advantage is too much, lose measure
  - this is where majority rule + weighting scheme is used
More details:
Assume claim, prove theorem:
Assume haven't terminated at \(i_0\)th stage

- so error \(C_{i_0}^x\) \(\leq \varepsilon\)

\[
\sum_x M_{i_0}(x) \geq \varepsilon 2^n
\]

Claim \(\Rightarrow\)

\[
\sum_x A_{i_0+1}(x) = \sum_x \sum_{0 \leq j \leq i_0} R_j(x) M_{i_0}(x) \quad \text{def of } A_{i_0+1}
\]

\[
= \sum_{0 \leq j \leq i_0} \text{Adv}_{C_{j+1}}(M_j) \quad \text{def of } \text{Adv}_{C_{j+1}}
\]

\[
= \left(2^n \varepsilon \right) (i_0 + 1)
\]

From "note"

\[
+ \sum_x A_{i_0+1}(x) \leq \sum_x \left( \frac{1}{\varepsilon Y} + \frac{\varepsilon Y}{2} (i_0 + 1) \right) \quad \text{claim}
\]

\[
= 2^n \left( \frac{1}{\varepsilon Y} + \frac{\varepsilon Y}{2} (i_0 + 1) \right)
\]

Putting together:

\[
(\varepsilon Y) (i_0 + 1) \leq \frac{1}{\varepsilon Y} + \frac{\varepsilon Y}{2} (i_0 + 1)
\]

so \(\varepsilon Y (i_0 + 1) \leq \frac{1}{\varepsilon Y} \Rightarrow i_0 \leq \frac{2}{\varepsilon^2 Y^2} - 1\)
Proof of claim:

Question: how can an input add to cumulative advantage throughout algorithm?

Observations:

- if algorithm's hypotheses $h_1...h_n$ are overwhelmingly correct on $x$, then not at all because $x$ gets measure 0

- if algorithm's hypotheses are doing badly (mostly wrong) then not at all because they decrease advantage

Main issue:

can wander in middle - majority correct but not large majority so have positive measure

increase advantage need to bound this case.
Proof of Claim

First, strange but obvious fact:

Fact "elevator argument"
If one spends any amount of time in an
elevator, then no matter what sequence of
buttons pushed, one ascends from $k^{th}$ to
$k^{1st}$ floor at most one more time
than one descends from the $k^{th}$ to $k^{th}$ floor.
(analogous for negative floors $-k$ to $-(k+1)$)

Proof by picture:

match transitions going up with
these going down on same level (as in parentheses)
but what is behavior of \( \sum_{j \in k} R(x) M_j(x) \)?

\[ \pm 1 \in [0, 1] \]

slope \( \leq 1 \) (in fact \( \leq 2 \))

same sign as \( N_j(x) \)

Recall: \( A_{\pm} = \sum_{j \in k} R_{\pm j} M_j(x) \)

Matching:

For \( k \geq 0 \):

match \( a = j \) st. \( N_j(x) = k \) \& \( N_{j+1}(x) = k+1 \)

with \( b = j' \) st. \( N_j'(x) = k' \) \& \( N_{j+1}'(x) = k' \)

For \( k < 0 \): analogously match \(-k\) to \((-k')\)

with \(-k'\) to \(-k\)

For each matched pair:

Will bound contribution from matched pairs

by \( \varepsilon x \) per pair using bound on slope

(and total of \( \frac{\varepsilon x \lambda}{2} \))
(for each matched pair \((a, b)\) cont.)

\[
R_{can}(x) M_a(x) + R_{cbh}(x) M_b(x) = M_a(x) - M_b(x)
\]

\[
\begin{array}{c}
N_a(x) = k \\
elevator going up
\end{array}
\quad
\begin{array}{c}
N_b(x) = k+1 \\
elevator going down
\end{array}
\]

if \(0 \leq K \leq \frac{1}{\varepsilon_Y}\) or \(0 \leq KH \leq \frac{1}{\varepsilon_Y}\)

then

\[
M_a(x) - M_b(x)
\]

\[
= (1 - \varepsilon_Y N_a(x)) - (1 - \varepsilon_Y N_b(x))
\]

\[
= K \varepsilon_X Y - KH \varepsilon_Y
\]

\[
= \varepsilon_Y
\]

else \(M_a(x) - M_b(x) = \begin{cases} 1 \text{ or } 0 \end{cases} = 0\)

\[\text{each pair contributes } \varepsilon Y \text{ to sum } \sum \leq \frac{1}{2} \text{ pairs}\]

\[\text{total contribution } \sum \frac{1}{2} \varepsilon_Y \]
Contribution from unmatched edges:

either all unmatched \( N_i \)'s have negative steps
or all have positive steps

if all negative:

\( R_{ij} \)’s all -1
\( M_j \)'s all \( \in [0,1] \)

.. contribution of \( R_{ij}(x)M_j(x) \leq 0 \)

if all positive:

if unmatched \( N_i \)'s in \( [0, \varepsilon x] \)

- for each \( M_j \in [0,1] \), contribution of

\( R_{ij}(x)M_j(x) \leq 1 \)

- at most \( \frac{1}{\varepsilon x} \) of these

\( \Rightarrow \) total contribution \( \leq \frac{1}{\varepsilon x} \)

if unmatched \( N_i \) in \( [\frac{\varepsilon x}{2}, \ldots] \)

then \( M_j = 0 \)

\( \Rightarrow \) total contribution = 0

\( \therefore \) Grand total \( \leq \frac{1}{2} \cdot \varepsilon \cdot x + \frac{1}{\varepsilon x} \)