Lecture 25: Algorithmic Learning Theory

Michael Dinitz

November 30, 2021
601.433/633 Introduction to Algorithms
Introduction

Machine Learning from the point of view of theoretical computer science
- Proofs about performance
- Minimize assumptions
- *Not* going to talk about useful in practice, etc.

Today:
- Concept Learning
- Online Learning
Concept Learning
Concept Learning Intro

Trying to learn “Yes/No” labels
  ▶ Given a photo, does it have a dog in it?
  ▶ Given an email, is it spam?

Given some labeled data. Create a good prediction rule (hypothesis) for future data.
Concept Learning Intro

Trying to learn “Yes/No” labels

- Given a photo, does it have a dog in it?
- Given an email, is it spam?

Given some labeled data. Create a good prediction rule (hypothesis) for future data.

Example: spam

- Want to create a rule (hypothesis) that will tell us whether an email is spam
- Given some example emails with labels (Yes / No, Spam / Not Spam)
### Example

<table>
<thead>
<tr>
<th>sales</th>
<th>size</th>
<th>Mr.</th>
<th>bad spelling</th>
<th>known-sender</th>
<th>spam?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

Reasonable hypothesis: spam if not known-sender AND (size OR sales)
### Example

<table>
<thead>
<tr>
<th>sales</th>
<th>size</th>
<th>Mr.</th>
<th>bad spelling</th>
<th>known-sender</th>
<th>spam?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

Reasonable hypothesis:
spam if not known-sender AND (size OR sales)
Questions

**Question 1:** Can we efficiently find working hypothesis for given labeled data?
- Mainly about efficiency; like many of the problems we’ve talked about
- Depends on what kinds of hypotheses we’re looking for (structure and quality)

**Question 2:** Can we be confident that our hypothesis will do well in the future?
- Not primarily about efficiency; about quality
- Requires knowing something about the future!
- Core of machine learning: use the past to make predictions about the future
Formalization: Beginning

Given sample set $S = \{(x^1, y^1), \ldots, (x^m, y^m)\}$. Size $m$ called the sample complexity

- Each $x^i$ drawn from distribution $D$ (not necessarily known)
- $y^i = f(x^i)$ for some unknown $f$

Our goal: compute hypothesis $h$ with low error on $D$:

$$\text{err}(h) := \Pr_{x \sim D}[h(x) \neq f(x)] \leq \epsilon$$
Formalization: Beginning

Given sample set \( S = \{(x^1, y^1), \ldots, (x^m, y^m)\} \). Size \( m \) called the \textit{sample complexity}

- Each \( x^i \) drawn from distribution \( D \) (not necessarily known)
- \( y^i = f(x^i) \) for some unknown \( f \)

Our goal: compute hypothesis \( h \) with low \textit{error} on \( D \):

\[
\text{err}(h) := \Pr_{x \sim D}[h(x) \neq f(x)] \leq \epsilon
\]

Generally not possible unless \( m \) extremely large. Proof: random function \( f \)

- Knowing \( f(x^i) \) on sample points doesn’t tell us anything about \( f(x) \) on points not sampled
Formalization: Beginning

Given sample set $S = \{(x^1, y^1), \ldots, (x^m, y^m)\}$. Size $m$ called the *sample complexity*

- Each $x^i$ drawn from distribution $D$ (not necessarily known)
- $y^i = f(x^i)$ for some unknown $f$

Our goal: compute hypothesis $h$ with low *error* on $D$:

$$\text{err}(h) := \Pr_{x \sim D}[h(x) \neq f(x)] \leq \epsilon$$

Generally not possible unless $m$ extremely large. Proof: random function $f$

- Knowing $f(x^i)$ on sample points doesn’t tell us anything about $f(x)$ on points not sampled

Need to restrict $f$. 
Example: Decision Lists

Data point: \( x \in \{0, 1\}^n \)

Decision List:
- If \( x_1 = 1 \) return 0
- Else if \( x_4 = 1 \) return 1
- Else if \( x_2 = 0 \) return 1
- Else return 0

Key features:
- Doesn’t branch
- Each “if” looks at one coordinate and either returns or continues down list
Example: Decision Lists

Data point: \( x \in \{0, 1\}^n \)

Decision List:
- If \( x_1 = 1 \) return 0
- Else if \( x_4 = 1 \) return 1
- Else if \( x_2 = 0 \) return 1
- Else return 0

Key features:
- Doesn’t branch
- Each “if” looks at one coordinate and either returns or continues down list

Can we “learn” decision lists? Restrict \( f \) to be a DL.
Example: Decision Lists

Data point: \( x \in \{0, 1\}^n \)

Decision List:

- If \( x_1 = 1 \) return 0
- Else if \( x_4 = 1 \) return 1
- Else if \( x_2 = 0 \) return 1
- Else return 0

Key features:

- Doesn’t branch
- Each “if” looks at one coordinate and either returns or continues down list

Can we “learn” decision lists? Restrict \( f \) to be a DL.

**Question 1:** Given sample data points labeled by some decision list, can we find a decision list that correctly labels the sample?

**Question 2:** Can we give an error bound with respect to distribution \( D \) that samples come from?
Formalization

Definition

Let $X$ be a collection of instances / data points (e.g., $X = \{0, 1\}^n$). A concept is a boolean function $h : X \to \{0, 1\}$ (e.g., a decision list), and a concept class $\mathcal{H}$ is a collection of concepts (e.g., all DLs).
Formalization

Definition

Let \( X \) be a collection of instances / data points (e.g., \( X = \{0, 1\}^n \)). A concept is a boolean function \( h : X \rightarrow \{0, 1\} \) (e.g., a decision list), and a concept class \( \mathcal{H} \) is a collection of concepts (e.g., all DLs).

Let \( m : \mathbb{R}^2 \rightarrow \mathbb{N} \).

Definition

A concept class \( \mathcal{H} \) is PAC-learnable with sample complexity \( m(\epsilon, \delta) \) if there is an algorithm \( A \) such that for all \( f \in \mathcal{H} \):

1. Input of \( A \) is \( 0 < \epsilon < 1/2 \) and \( 0 < \delta < 1/2 \) and set \( S = \{(x^1, y^1), \ldots, (x^m(\epsilon, \delta), y^m(\epsilon, \delta))\} \) where \( y^i = f(x^i) \) for all \( i \)
2. \( A \) outputs a concept \( h \) that is “probably approximately correct”:

\[
\Pr_{S \sim D^m(\epsilon, \delta)}[\text{err}(h) \leq \epsilon] = \Pr_{S \sim D^m(\epsilon, \delta)}\left[\Pr_{x \sim D}[h(x) \neq f(x)] \leq \epsilon\right] \geq 1 - \delta
\]
Learning Decision Lists

Are decision lists PAC-learnable with low sample complexity and efficient algorithms?
Learning Decision Lists

Are decision lists PAC-learnable with low sample complexity and efficient algorithms?

\[ S' = S, L = \emptyset \]

while \( S' \neq \emptyset \) {

Find if-then rule \( \alpha \) consistent with \( S' \) that labels at least 1 element of \( S' \)

Add \( \alpha \) to the bottom of \( L \)

Remove data labeled by \( \alpha \) from \( S' \)

}

Add “else return 0” to bottom of \( L \)

Return \( L \)

Correctness: Why always finds such a rule?

By assumption, there is a DL \( f \) that labels \( S \) and so \( S' \neq \emptyset \)

Highest rule in \( f \) not added to \( L \) will work!
Learning Decision Lists

Are decision lists PAC-learnable with low sample complexity and efficient algorithms?

\[
S' = S, \ L = \emptyset \\
\text{while}(S' \neq \emptyset) \ { \\
\quad \text{Find if-then rule } \alpha \text{ consistent with } S' \text{ that labels at least 1 element of } S' \\
\quad \text{Add } \alpha \text{ to the bottom of } L \\
\quad \text{Remove data labeled by } \alpha \text{ from } S' \\
}\}
\]

Add “else return 0” to bottom of L

Return L

Correctness: Why can we always find such an \( \alpha \)?
Learning Decision Lists

Are decision lists PAC-learnable with low sample complexity and efficient algorithms?

\[
S' = S, \ L = \emptyset
\]

while \((S' \neq \emptyset)\) {
  Find if-then rule \(\alpha\) consistent with \(S'\) that labels at least 1 element of \(S'\)
  Add \(\alpha\) to the bottom of \(L\)
  Remove data labeled by \(\alpha\) from \(S'\)
}

Add “else return 0” to bottom of \(L\)

Return \(L\)

**Correctness**: Why can we always find such an \(\alpha\)?

- By assumption, there is a DL \(f\) that labels \(S\) and so \(S'\)
- Highest rule in \(f\) not added to \(L\) will work!
Running Time of Algorithm

Number of iterations: $\leq |S| = m(\epsilon, \delta)$

Time per iteration: check every possible rule, see if consistent with $S'$ (and labels at least one point)

Number of possible rules ("if $x_i = 0 \implies 1$", return 0): $4n$

Total time at most $O(n \cdot m(\epsilon, \delta))$: pretty good if sample complexity small.

Sample Complexity: We are worried about outputting DL $h$ with $\text{err}(h) > \epsilon$: want this to happen with probability at most $\delta$.

But the DL $H$ we output labels $S$ correctly!

Want to show: since $h$ labels $S$ correctly, with probability at least $1 - \delta$, has error at most $\epsilon$.

In other words: prove that with probability at least $1 - \delta$, every DL $h$ consistent with $S$ has error at most $\epsilon$. 

Michael Dinitz

Lecture 25: Algorithmic Learning Theory

November 30, 2021 11 / 21
Running Time of Algorithm

Number of iterations: $\leq |S| = m(\epsilon, \delta)$

Time per iteration: check every possible rule, see if consistent with $S'$ (and labels at least one point)
- Number of possible rules ("if $x_i = 0/1$, return $0/1$"): $4n$
Running Time of Algorithm

Number of iterations: \( \leq |S| = m(\epsilon, \delta) \)

Time per iteration: check every possible rule, see if consistent with \( S' \) (and labels at least one point)

- Number of possible rules ("if \( x_i = 0/1 \), return 0/1"): \( 4n \)

Total time at most \( O(n \cdot m(\epsilon, \delta)) \): pretty good if sample complexity small.
Running Time of Algorithm

Number of iterations: $|\mathcal{S}| = m(\epsilon, \delta)$

Time per iteration: check every possible rule, see if consistent with $\mathcal{S}'$ (and labels at least one point)
  - Number of possible rules (“if $x_i = 0/1$, return 0/1”): $4n$

Total time at most $O(n \cdot m(\epsilon, \delta))$: pretty good if sample complexity small.

Sample Complexity: We are worried about outputting DL $h$ with $\text{err}(h) > \epsilon$: want this to happen with probability at most $\delta$.
  - But the DL $H$ we output labels $\mathcal{S}$ correctly!
  - Want to show: since $h$ labels $\mathcal{S}$ correctly, with probability at least $1 - \delta$ has error at most $\epsilon$
  - In other words: prove that with probability at least $1 - \delta$, every DL $h$ consistent with $\mathcal{S}$ has error at most $\epsilon$
Sample Complexity
So suppose that $h$ some DL with error at least $\epsilon$ ($\Pr_{x \sim D}[h(x) \neq f(x)] \geq \epsilon$), and let $m = m(\epsilon, \delta) = |S|$
Sample Complexity

So suppose that \( h \) some DL with error at least \( \epsilon \) \( (\Pr_{x \sim D}[h(x) \neq f(x)] \geq \epsilon) \), and let \( m = m(\epsilon, \delta) = |S| \)

\[ \implies \Pr_{S \sim D^m}[h \text{ consistent with } S] \leq (1 - \epsilon)^m \]
Sample Complexity

So suppose that \( h \) some DL with error at least \( \epsilon \) (\( \Pr_{x \sim D}[h(x) \neq f(x)] \geq \epsilon \)), and let \( m = m(\epsilon, \delta) = |S| \)

\[ \implies \Pr_{S \sim D^m}[h \text{ consistent with } S] \leq (1 - \epsilon)^m \]

Let \( H = \# \text{ decision lists.} \)

\[ \Pr_{S \sim D^m}[\exists h \text{ s.t. } \text{err}(h) > \epsilon, h \text{ consistent with } S] \leq H(1 - \epsilon)^m \leq He^{-\epsilon m} \]
Sample Complexity

So suppose that \( h \) some DL with error at least \( \epsilon (\Pr_{x \sim D}[h(x) \neq f(x)] \geq \epsilon) \), and let
\[
m = m(\epsilon, \delta) = |S|
\]

\[\implies \Pr_{S \sim D^m}[h \text{ consistent with } S] \leq (1 - \epsilon)^m\]

Let \( H = \# \) decision lists.

\[\Pr_{S \sim D^m}[\exists h \text{ s.t. err}(h) > \epsilon, h \text{ consistent with } S] \leq H(1 - \epsilon)^m \leq He^{-\epsilon m}\]

Set \( m = \frac{1}{\epsilon} \left( \ln H + \ln \left( \frac{1}{\delta} \right) \right) \):

\[= He^{-\epsilon m} \leq He^{-\epsilon \left( \frac{1}{\epsilon} \left( \ln |H| + \ln \left( \frac{1}{\delta} \right) \right) \right)} = He^{-\left( \ln |H| + \ln \left( \frac{1}{\delta} \right) \right)} = H \left( \frac{1}{H} \right) \delta = \delta\]
Sample Complexity

So suppose that $\mathbf{h}$ some DL with error at least $\epsilon \left( \Pr_{x \sim D}[h(x) \neq f(x)] \geq \epsilon \right)$, and let

$m = m(\epsilon, \delta) = |S|$

$\implies \Pr_{S \sim D^m}[\exists h \text{ consistent with } S] \leq (1 - \epsilon)^m$

Let $H = \#$ decision lists.

$$\Pr_{S \sim D^m}[\exists h \text{ s.t. } \text{err}(h) > \epsilon, h \text{ consistent with } S] \leq H(1 - \epsilon)^m \leq H e^{-\epsilon m}$$

Set $m = \frac{1}{\epsilon} \left( \ln H + \ln \left( \frac{1}{\delta} \right) \right)$:

$$= H e^{-\epsilon m} \leq H e^{-\epsilon \frac{1}{\epsilon} \left( \ln |H| + \ln \left( \frac{1}{\delta} \right) \right)} = H e^{-\left( \ln |H| + \ln \left( \frac{1}{\delta} \right) \right)} = H \left( \frac{1}{H} \right) \delta = \delta$$

So with probability at least $1 - \delta$, every DL consistent with $S$ has error at most $\epsilon$ (including the one we output)!
Sample Complexity

So suppose that $h$ some DL with error at least $\epsilon$ ($\Pr_{x \sim D}[h(x) \neq f(x)] \geq \epsilon$), and let $m = m(\epsilon, \delta) = |S|$

$$\implies \Pr_{S \sim D^m}[h \text{ consistent with } S] \leq (1 - \epsilon)^m$$

Let $H = \#$ decision lists.

$$\Pr_{S \sim D^m}[\exists h \text{ s.t. } \text{err}(h) > \epsilon, h \text{ consistent with } S] \leq H(1 - \epsilon)^m \leq H e^{-\epsilon m}$$

Set $m = \frac{1}{\epsilon} \left( \ln H + \ln \left( \frac{1}{\delta} \right) \right)$:

$$= He^{-\epsilon m} \leq He^{-\frac{1}{\epsilon} \left( \ln |H| + \ln \left( \frac{1}{\delta} \right) \right)} = He^{-\left( \ln |H| + \ln \left( \frac{1}{\delta} \right) \right)} = H \left( \frac{1}{H} \right) \delta = \delta$$

So with probability at least $1 - \delta$, every DL consistent with $S$ has error at most $\epsilon$ (including the one we output)!

$H \leq n!4^n$, since at most $n!$ orderings of coordinates, and at most 4 rules/coordinate

$$\implies m = \Theta \left( \frac{1}{\epsilon} \left( n \ln n + \ln \left( \frac{1}{\delta} \right) \right) \right)$$
Occam’s Razor

“Prefer simple explanations to complicated ones”

Only thing we used about DL in sample complexity analysis: \( H \leq n!4^n \)

“Simple” hypothesis: expressible in \( \leq s \) bits
Occam’s Razor

“Prefer simple explanations to complicated ones”

Only thing we used about DL in sample complexity analysis: $H \leq n!4^n$

“Simple” hypothesis: expressible in $\leq s$ bits

$\implies \leq 2^s$ simple hypotheses
Occam’s Razor

“Prefer simple explanations to complicated ones”

Only thing we used about DL in sample complexity analysis: $H \leq n!4^n$

“Simple” hypothesis: expressible in $\leq s$ bits

$\implies \leq 2^s$ simple hypotheses

$\implies$ after $\frac{1}{\epsilon} \left( s \ln 2 + \ln \left( \frac{1}{\delta} \right) \right)$ samples, unlikely for us to get fooled by a simple hypothesis that’s actually wrong!
Online Learning
Online Learning

Learning over time, not just one-shot

- Similar to online algorithms: see data one piece at a time
- Instead of trying to minimize competitive ratio, trying to use the data to make decisions as we go.

Remove assumption that $D$ fixed
Learning From Expert Advice

Intuition: stock market

- $n$ experts
- Every day:
  - Every expert predicts up/down
  - Algorithm makes prediction
  - Find out what happened

What can/should we do? Can we always make an accurate prediction?
Learning From Expert Advice

Intuition: stock market

- $n$ experts
- Every day:
  - Every expert predicts up/down
  - Algorithm makes prediction
  - Find out what happened

What can/should we do? Can we always make an accurate prediction?

- No! Experts could all be essentially random, uncorrelated with market
Learning From Expert Advice

Intuition: stock market

- \( n \) experts
- Every day:
  - Every expert predicts up/down
  - Algorithm makes prediction
  - Find out what happened

What can/should we do? Can we always make an accurate prediction?

- No! Experts could all be essentially random, uncorrelated with market

Easier (but still interesting) goal: can we do as well as the best expert?

- Don’t try to learn the market: learn which expert knows the market best
Warmup

Assume best expert makes 0 mistakes: always correctly predicts the market. How should we predict market to minimize #mistakes?
Warmup

Assume best expert makes 0 mistakes: always correctly predicts the market. How should we predict market to minimize \#mistakes?

Each day:
Warmup

Assume best expert makes 0 mistakes: always correctly predicts the market. How should we predict market to minimize #mistakes?

Each day:
- Majority vote of remaining experts
- Remove incorrect experts
Warmup

Assume best expert makes 0 mistakes: always correctly predicts the market. How should we predict market to minimize #mistakes?

Each day:
- Majority vote of remaining experts
- Remove incorrect experts

Best expert makes 0 mistakes

We make:
Warmup

Assume best expert makes 0 mistakes: always correctly predicts the market. How should we predict market to minimize #mistakes?

Each day:
- Majority vote of remaining experts
- Remove incorrect experts

Best expert makes 0 mistakes
We make: $O(\log n)$ mistakes
Assume best expert makes $0$ mistakes: always correctly predicts the market. How should we predict market to minimize $\#$ mistakes?

Each day:
- Majority vote of remaining experts
- Remove incorrect experts

Best expert makes $0$ mistakes

We make: $O(\log n)$ mistakes
- Each mistake decreases $\#$ experts by $1/2$
General case: no perfect expert

Weighted Majority

Initialize all experts to weight $\frac{1}{2}$

Predict based on weighted majority vote

Penalize mistakes by cutting weights in half

$W = \text{total weight}$

$W \geq \left( \frac{1}{2} \right)^M$

Best expert has weight at least $\left( \frac{1}{2} \right)^M$

$W \leq n \left( \frac{3}{4} \right)^M$

Every time we make a mistake, at least $\frac{1}{2}$ the total weight gets decreased by $\frac{1}{2}$, so left with at most $\frac{3}{4}$ of the original total weight

$\Rightarrow \left( \frac{1}{2} \right)^M \leq n \left( \frac{3}{4} \right)^M$

$\Rightarrow \left( \frac{4}{3} \right)^M \leq n^2 M$

$\Rightarrow M \leq \log_4 \left( \frac{3}{4} \right) (n^2 M)$

$\approx 2^{m + \log n}$

Michael Dinitz
Lecture 25: Algorithmic Learning Theory
November 30, 2021 18 / 21
General case: no perfect expert

Weighted Majority
- Initialize all experts to weight 1
- Predict based on *weighted* majority vote
- Penalize mistakes by cutting weights in half

\[ M = \# \text{ mistakes we've made} \]
\[ m = \# \text{ mistakes best expert has made} \]
\[ W = \text{ total weight} \]
General case: no perfect expert

Weighted Majority

- Initialize all experts to weight 1
- Predict based on weighted majority vote
- Penalize mistakes by cutting weights in half

\[ M = \# \text{ mistakes we've made} \]
\[ m = \# \text{ mistakes best expert has made} \]
\[ W = \text{ total weight} \]

\[ W \geq 1/2^m \]

- Best expert has weight at least \( (1/2)^m \)
General case: no perfect expert

Weighted Majority
- Initialize all experts to weight 1
- Predict based on weighted majority vote
- Penalize mistakes by cutting weights in half

\( M = \# \text{ mistakes we've made} \)
\( m = \# \text{ mistakes best expert has made} \)
\( W = \text{ total weight} \)

\[ W \geq (1/2)^m \]
- Best expert has weight at least \((1/2)^m\)

\[ W \leq n(3/4)^M \]
- Every time we make a mistake, at least 1/2 the total weight gets decreased by 1/2, so left with at most 3/4 of the original total weight

Michael Dinitz
Lecture 25: Algorithmic Learning Theory
November 30, 2021 18 / 21
General case: no perfect expert

Weighted Majority

- Initialize all experts to weight 1
- Predict based on weighted majority vote
- Penalize mistakes by cutting weights in half

\[ M = \# \text{ mistakes we’ve made} \]
\[ m = \# \text{ mistakes best expert has made} \]
\[ W = \text{ total weight} \]

\[ W \geq (1/2)^m \]

- Best expert has weight at least \((1/2)^m\)

\[ \implies (1/2)^m \leq n (3/4)^M \]
\[ \implies (4/3)^M \leq n^{2^m} \]
\[ \implies M \leq \log_{4/3} (n^{2^m}) = \frac{m + \log n}{\log (4/3)} \approx 2.4 (m + \log n) \]
Improved Algorithm

How to do better?

Randomization!

Let $W_i = 1$ be weight of expert $i$, let $W = \sum_{i=1}^{n} W_i$.

Do what expert $i$ says with probability $W_i / W$.

If expert $i$ incorrect, set $W_i \leftarrow (1 - \varepsilon) W_i$.

Theorem

Let $M =$ mistakes we've made, let $m =$ mistakes best expert made.

When $\varepsilon \leq \frac{1}{2}$:

$$E[M] \leq \left(1 + \varepsilon\right)m + \frac{1}{\varepsilon} \ln n$$
Improved Algorithm

How to do better? Randomization!

Let $W_i = 1$ be the weight of expert $i$, let $W = \sum_{i=1}^{n} W_i$.

- Do what expert $i$ says with probability $W_i / W$.
- If expert $i$ is incorrect, set $W_i \leftarrow (1 - \epsilon) W_i$.

Theorem

Let $M =$ number of mistakes we've made, let $m =$ number of mistakes by the best expert.

When $\epsilon \leq 1/2$:

$$\mathbb{E}[M] \leq (1 + \epsilon) m + \frac{1}{\epsilon} \ln n.$$
Improved Algorithm

How to do better? Randomization! (and change $1/2$ to $(1 - \epsilon)$)
Improved Algorithm

How to do better? Randomization! (and change $1/2$ to $(1 - \epsilon)$)

**Randomized** Weighted Majority
- Let $W_i = 1$ be weight of expert $i$, let $W = \sum_{i=1}^{n} W_i$.
- Do what expert $i$ says with probability $W_i/W$
- If expert $i$ incorrect, set $W_i \leftarrow (1 - \epsilon)W_i$
**Improved Algorithm**

How to do better? Randomization! (and change $1/2$ to $(1 - \epsilon)$)

**Randomized Weighted Majority**

- Let $W_i = 1$ be weight of expert $i$, let $W = \sum_{i=1}^{n} W_i$.
- Do what expert $i$ says with probability $W_i/W$.
- If expert $i$ incorrect, set $W_i \leftarrow (1 - \epsilon)W_i$.

**Theorem**

Let $M = \#$ mistakes we’ve made, let $m = \#$ mistakes best expert has made.

When $\epsilon \leq 1/2$:

$$E[M] \leq (1 + \epsilon)m + \frac{1}{\epsilon} \ln n$$
Randomized Weighted Majority Analysis

Let:

- \( F_i \) = fraction of weight at time \( i \) on experts who make mistake at time \( i \)
- \( W_i \) = total weight after time \( i \) (at beginning of time \( i + 1 \))
Randomized Weighted Majority Analysis

Let:

- $F_i =$ fraction of weight at time $i$ on experts who make mistake at time $i$
- $W_i =$ total weight *after* time $i$ (at beginning of time $i+1$)

\[ W_0 = n \]
Randomized Weighted Majority Analysis

Let:

- \( F_i \) = fraction of weight at time \( i \) on experts who make mistake at time \( i \)
- \( W_i \) = total weight after time \( i \) (at beginning of time \( i + 1 \))

\[
\begin{align*}
W_0 &= n \\
W_1 &= F_1 W_0 (1 - \epsilon) + (1 - F_1) W_0 = F_1 n (1 - \epsilon) + (1 - F_1) n \\
&= n (F_1 - \epsilon F_1 + 1 - F_1) = (1 - \epsilon F_1) n
\end{align*}
\]
Randomized Weighted Majority Analysis

Let:

- \( F_i \) = fraction of weight at time \( i \) on experts who make mistake at time \( i \)
- \( W_i \) = total weight after time \( i \) (at beginning of time \( i + 1 \))

\[
W_0 = n \\
W_1 = F_1 W_0 (1 - \epsilon) + (1 - F_1) W_0 = F_1 n (1 - \epsilon) + (1 - F_1) n \\
= n (F_1 - \epsilon F_1 + 1 - F_1) = (1 - \epsilon F_1) n \\
W_2 = F_2 W_1 (1 - \epsilon) + (1 - F_2) W_1 = (1 - \epsilon F_2) W_1 = (1 - \epsilon F_2) (1 - \epsilon) F_1 n
\]
Randomized Weighted Majority Analysis

Let:
- $F_i =$ fraction of weight at time $i$ on experts who make mistake at time $i$
- $W_i =$ total weight after time $i$ (at beginning of time $i + 1$)

\[
\begin{align*}
W_0 &= n \\
W_1 &= F_1 W_0 (1 - \epsilon) + (1 - F_1) W_0 = F_1 n (1 - \epsilon) + (1 - F_1) n \\
    &= n (F_1 - \epsilon F_1 + 1 - F_1) = (1 - \epsilon F_1) n \\
W_2 &= F_2 W_1 (1 - \epsilon) + (1 - F_2) W_1 = (1 - \epsilon F_2) W_1 = (1 - \epsilon F_2) (1 - \epsilon) F_1 n \\
    &\vdots \\
W_t &= n \prod_{i=1}^{t} (1 - \epsilon F_i) \leq n \prod_{i=1}^{t} e^{-\epsilon F_i} = ne^{-\epsilon \sum_{i=1}^{t} F_i}
\end{align*}
\]
Randomized Weighted Majority Analysis (cont’d)

Note: probability we make mistake at time \( i \) is exactly \( F_i \) \( \implies \) \( E[M] = \sum_{i=1}^{t} F_i \)
Randomized Weighted Majority Analysis (cont’d)

Note: probability we make mistake at time $i$ is exactly $F_i \implies E[M] = \sum_{i=1}^{t} F_i$

\[ \implies \ln W_t \leq \ln \left( ne^{-\epsilon \sum_{i=1}^{t} F_i} \right) = \ln n - \epsilon \sum_{i=1}^{t} F_i = \ln n - \epsilon E[M] \]
Randomized Weighted Majority Analysis (cont’d)

Note: probability we make mistake at time \( i \) is exactly \( F_i \) \( \implies \) \( E[M] = \sum_{i=1}^{t} F_i \)

\[
\ln W_t \leq \ln \left( ne^{-\varepsilon \sum_{i=1}^{t} F_i} \right) = \ln n - \varepsilon \sum_{i=1}^{t} F_i = \ln n - \varepsilon E[M]
\]

But best expert makes \( m \) mistakes

\[
\implies W_t \geq (1 - \varepsilon)^m \implies \ln W_t \geq m \ln(1 - \varepsilon)
\]
Randomized Weighted Majority Analysis (cont’d)

Note: probability we make mistake at time \( i \) is exactly \( F_i \)

\[ \implies E[M] = \sum_{i=1}^{t} F_i \]

\[ \implies \ln W_t \leq \ln \left( n e^{-\epsilon \sum_{i=1}^{t} F_i} \right) = \ln n - \epsilon \sum_{i=1}^{t} F_i = \ln n - \epsilon E[M] \]

But best expert makes \( m \) mistakes

\[ \implies W_t \geq (1 - \epsilon)^m \implies \ln W_t \geq m \ln(1 - \epsilon) \]

So \( m \ln(1 - \epsilon) \leq \ln n - \epsilon E[M] \)
Randomized Weighted Majority Analysis (cont’d)

Note: probability we make mistake at time $i$ is exactly $F_i \implies \mathbb{E}[M] = \sum_{i=1}^{t} F_i$

$$\implies \ln W_t \leq \ln \left( ne^{-\varepsilon \sum_{i=1}^{t} F_i} \right) = \ln n - \varepsilon \sum_{i=1}^{t} F_i = \ln n - \varepsilon \mathbb{E}[M]$$

But best expert makes $m$ mistakes

$$\implies W_t \geq (1 - \varepsilon)^m \implies \ln W_t \geq m \ln(1 - \varepsilon)$$

So $m \ln(1 - \varepsilon) \leq \ln n - \varepsilon \mathbb{E}[M]$

$$\implies \mathbb{E}[M] \leq \frac{1}{\varepsilon} \left( \ln n - m \ln(1 - \varepsilon) \right) \leq (1 + \varepsilon)m + \frac{1}{\varepsilon} \ln n$$

(using fact that $\frac{-\ln(1-\varepsilon)}{\varepsilon} \leq 1 + \varepsilon$ for all $0 < \varepsilon \leq 1/2$)