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

sales	size	Mr.	bad spelling	known-sender	spam?
Y	N	Y	Y	N	Y
N	N	N	Y	Y	N
N	Y	N	N	N	Y
Y	N	N	N	Y	N
N	N	Y	N	Y	N
Y	N	N	Y	N	Y
N	N	Y	N	N	N
N	Y	N	Y	N	Y

Example

sales	size	Mr.	bad spelling	known-sender	spam?
Y	N	Y	Y	N	Y
N	N	N	Y	Y	N
N	Y	N	N	N	Y
Y	N	N	N	Y	N
N	N	Y	N	Y	N
Y	N	N	Y	N	Y
N	N	Y	N	N	N
N	Y	N	Y	N	Y

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

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

Our goal: compute hypothesis $\mathbf{h}$ with low *error* on $\mathbf{D}$:

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

Formalization: Beginning

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

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

Our goal: compute hypothesis $\mathbf{h}$ with low *error* on $\mathbf{D}$:

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

Generally not possible unless $\mathbf{m}$ extremely large. Proof: random function $\mathbf{f}$

- Knowing $\mathbf{f}(\mathbf{x}^i)$ on sample points doesn't tell us anything about $\mathbf{f}(\mathbf{x})$ on points not sampled

Formalization: Beginning

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

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

Our goal: compute hypothesis $\mathbf{h}$ with low *error* on $\mathbf{D}$:

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

Generally not possible unless $\mathbf{m}$ extremely large. Proof: random function $\mathbf{f}$

- Knowing $\mathbf{f}(\mathbf{x}^i)$ on sample points doesn't tell us anything about $\mathbf{f}(\mathbf{x})$ on points not sampled

Need to restrict $\mathbf{f}$.

Example: Decision Lists

Data point: $\mathbf{x} \in \{0,1\}^n$

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: $\mathbf{x} \in \{0, 1\}^n$

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: $\mathbf{x} \in \{0,1\}^n$

Decision List:

- If $\mathbf{x_1} = 1$ return $\mathbf{0}$
- Else if $\mathbf{x_4} = 1$ return $\mathbf{1}$
- Else if $\mathbf{x_2} = 0$ return $\mathbf{1}$
- Else return $\mathbf{0}$

Key features:

- Doesn't branch
- Each "if" looks at one coordinate and either returns or continues down list

Can we "learn" decision lists? Restrict $\mathbf{f}$ 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 $\mathbf{D}$ that samples come from?

Formalization

Definition

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

Formalization

Definition

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

Let $\mathbf{m} : \mathbb{R}^2 \to \mathbb{N}$.

Definition

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

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

dot after

$$\Pr_{S \sim D^{m(\epsilon,\delta)}}[\mathrm{err}(h) \le \epsilon] = \Pr_{S \sim D^{m(\epsilon,\delta)}}\left[\Pr_{x \sim D}[h(x) \neq f(x)] \le \epsilon\right] \ge 1 - \delta$$

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?

$\mathbf{S}' = \mathbf{S}, \mathbf{L} = \varnothing$
while($\mathbf{S}' \neq \varnothing$) $\{$
    Find if-then rule α consistent with $\mathbf{S}'$ that labels at least $\mathbf{1}$ element of $\mathbf{S}'$
    Add α to the bottom of $\mathbf{L}$
    Remove data labeled by α from $\mathbf{S}'$
$\}$
Add "else return $\mathbf{0}$" to bottom of $\mathbf{L}$
Return $\mathbf{L}$

Learning Decision Lists

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

$\mathbf{S}' = \mathbf{S}, \mathbf{L} = \varnothing$
while($\mathbf{S}' \neq \varnothing$) {
    Find if-then rule α consistent with $\mathbf{S}'$ that labels at least $\mathbf{1}$ element of $\mathbf{S}'$
    Add α to the bottom of $\mathbf{L}$
    Remove data labeled by α from $\mathbf{S}'$
}
Add "else return $\mathbf{0}$" to bottom of $\mathbf{L}$
Return $\mathbf{L}$

Correctness: Why can we always find such an α?

Learning Decision Lists

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

$\mathbf{S}' = \mathbf{S}, \mathbf{L} = \varnothing$
while($\mathbf{S}' \neq \varnothing$) {
    Find if-then rule α consistent with $\mathbf{S}'$ that labels at least $\mathbf{1}$ element of $\mathbf{S}'$
    Add α to the bottom of $\mathbf{L}$
    Remove data labeled by α from $\mathbf{S}'$
}
Add "else return $\mathbf{0}$" to bottom of $\mathbf{L}$
Return $\mathbf{L}$

Correctness: Why can we always find such an α?

- By assumption, there is a DL $\mathbf{f}$ that labels $\mathbf{S}$ and so $\mathbf{S}'$
- Highest rule in $\mathbf{f}$ not added to $\mathbf{L}$ will work!

Running Time of Algorithm

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

Running Time of Algorithm

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

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

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

Running Time of Algorithm

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

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

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

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

Running Time of Algorithm

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

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

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

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

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

- But the DL $\mathbf{h}$ we output labels $\mathbf{S}$ correctly!
- Want to show: since $\mathbf{h}$ labels $\mathbf{S}$ correctly, with probability at least $\mathbf{1} - \delta$ has error at most ϵ
- In other words: prove that with probability at least $\mathbf{1} - \delta$, *every* DL $\mathbf{h}$ consistent with $\mathbf{S}$ has error at most ϵ

Sample Complexity

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

Sample Complexity

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

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

Sample Complexity

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

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

Let $\mathbf{H} = \#$ decision lists.

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

Sample Complexity

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

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

Let $\mathbf{H} = \#$ decision lists.

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

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

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

Sample Complexity

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

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

Let $\mathbf{H} = \#$ decision lists.

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

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

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

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

Sample Complexity

So suppose that $\mathbf{h}$ some DL with error at least ϵ ($\mathbf{Pr}_{\mathbf{x}\sim\mathbf{D}}[\mathbf{h}(\mathbf{x}) \neq \mathbf{f}(\mathbf{x})] \geq \epsilon$), and let
$\mathbf{m} = \mathbf{m}(\epsilon, \delta) = |\mathbf{S}|$
$\implies \mathbf{Pr}_{\mathbf{S}\sim\mathbf{D}^\mathbf{m}}[\mathbf{h}$ consistent with $\mathbf{S}] \leq (\mathbf{1} - \epsilon)^\mathbf{m}$

Let $\mathbf{H} = \#$ decision lists.

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

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

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

So with probability at least $\mathbf{1} - \delta$, *every* DL consistent with $\mathbf{S}$ has error at most ϵ (including the one we output)!
$\mathbf{H} \leq \mathbf{n}!\mathbf{4}^\mathbf{n}$, since at most $\mathbf{n}!$ orderings of coordinates, and at most $\mathbf{4}$ rules/coordinate
$\implies \mathbf{m} = \Theta\left(\frac{1}{\epsilon}\left(\mathbf{n}\ln\mathbf{n} + \ln\left(\frac{1}{\delta}\right)\right)\right)$

Occam's Razor

"Prefer simple explanations to complicated ones"

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

"Simple" hypothesis: expressible in $\leq \mathbf{s}$ bits

Occam's Razor

"Prefer simple explanations to complicated ones"

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

"Simple" hypothesis: expressible in $\leq \mathbf{s}$ bits

$\implies \leq \mathbf{2^s}$ simple hypotheses

Occam's Razor

"Prefer simple explanations to complicated ones"

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

"Simple" hypothesis: expressible in $\leq \mathbf{s}$ bits

$\implies \leq \mathbf{2^s}$ simple hypotheses

$\implies$ after $\frac{1}{\epsilon}\left(\mathbf{s}\ln 2 + \ln\left(\frac{1}{\delta}\right)\right)$ 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 $\mathbf{D}$ fixed

Learning From Expert Advice

Intuition: stock market

- $\mathbf{n}$ 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 $\mathbf{0}$ mistakes: always correctly predicts the market.
How should we predict market to minimize #mistakes?

Warmup

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

Each day:

Warmup

Assume best expert makes $\mathbf{0}$ 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 $\mathbf{0}$ 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 $\mathbf{0}$ mistakes

We make:

Warmup

Assume best expert makes $\mathbf{0}$ 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 $\mathbf{0}$ mistakes

We make: $\mathbf{O(\log n)}$ mistakes

Warmup

Assume best expert makes $\mathbf{0}$ 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 $\mathbf{0}$ mistakes

We make: $\mathbf{O(\log n)}$ mistakes

- Each mistake decreases # experts by $\mathbf{1/2}$

General case: no perfect expert

General case: no perfect expert

Weighted Majority

- Initialize all experts to weight $\mathbf{1}$
- Predict based on *weighted* majority vote
- Penalize mistakes by cutting weights in half

$\mathbf{M}$ = # mistakes we've made
$\mathbf{m}$ = # mistakes best expert has made
$\mathbf{W}$ = total weight

General case: no perfect expert

Weighted Majority

- Initialize all experts to weight $\mathbf{1}$
- Predict based on *weighted* majority vote
- Penalize mistakes by cutting weights in half

$\mathbf{M}$ = # mistakes we've made
$\mathbf{m}$ = # mistakes best expert has made
$\mathbf{W}$ = total weight

$\mathbf{W \geq (1/2)^m}$

- Best expert has weight at least $\mathbf{(1/2)^m}$

General case: no perfect expert

Weighted Majority

- Initialize all experts to weight $\mathbf{1}$
- Predict based on *weighted* majority vote
- Penalize mistakes by cutting weights in half

$\mathbf{M}$ = # mistakes we've made
$\mathbf{m}$ = # mistakes best expert has made
$\mathbf{W}$ = total weight

$\mathbf{W} \geq (1/2)^{\mathbf{m}}$

- Best expert has weight at least $(1/2)^{\mathbf{m}}$

$\mathbf{W} \leq \mathbf{n}(3/4)^{\mathbf{M}}$

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

General case: no perfect expert

Weighted Majority

- Initialize all experts to weight $\mathbf{1}$
- Predict based on *weighted* majority vote
- Penalize mistakes by cutting weights in half

$\mathbf{M}$ = # mistakes we've made
$\mathbf{m}$ = # mistakes best expert has made
$\mathbf{W}$ = total weight

$\mathbf{W \geq (1/2)^m}$

- Best expert has weight at least $\mathbf{(1/2)^m}$

$\mathbf{W \leq n(3/4)^M}$

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

$$\implies (1/2)^m \leq n(3/4)^M \implies (4/3)^M \leq n2^m$$

$$\implies M \leq \log_{4/3}(n2^m) = \frac{m + \log n}{\log(4/3)} \approx 2.4(m + \log n)$$

Improved Algorithm

How to do better?

Improved Algorithm

How to do better? Randomization!

Improved Algorithm

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

Improved Algorithm

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

Randomized Weighted Majority

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

Improved Algorithm

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

Randomized Weighted Majority

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

Theorem

Let $\mathbf{M}$ *= # mistakes we've made, let* $\mathbf{m}$ *= # mistakes best expert has made.*
When $\epsilon \leq 1/2$*:*

$$\mathbf{E}[\mathbf{M}] \leq (1+\epsilon)\mathbf{m} + \frac{1}{\epsilon}\ln n$$

Randomized Weighted Majority Analysis

Let:

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

Randomized Weighted Majority Analysis

Let:

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

$$\mathbf{W_0 = n}$$

Randomized Weighted Majority Analysis

Let:

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

$$\mathbf{W_0 = n}$$

$$\mathbf{W_1 = F_1 W_0 (1-\epsilon) + (1-F_1) W_0 = F_1 n (1-\epsilon) + (1-F_1) n}$$

$$\mathbf{= n(F_1 - \epsilon F_1 + 1 - F_1) = (1 - \epsilon F_1) n}$$

Randomized Weighted Majority Analysis

Let:

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

$$
\begin{aligned}
W_0 &= n \\
W_1 &= F_1 W_0 (1-\epsilon) + (1-F_1) W_0 = F_1 n (1-\epsilon) + (1-F_1) n \\
&= n(F_1 - \epsilon F_1 + 1 - F_1) = (1 - \epsilon F_1) n \\
W_2 &= F_2 W_1 (1-\epsilon) + (1-F_2) W_1 = (1-\epsilon F_2) W_1 = (1-\epsilon F_2)(1-\epsilon) F_1 n
\end{aligned}
$$

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 $i+1$)

$$
\begin{aligned}
W_0 &= n \\
W_1 &= F_1 W_0 (1-\epsilon) + (1-F_1) W_0 = F_1 n (1-\epsilon) + (1-F_1) n \\
&= n(F_1 - \epsilon F_1 + 1 - F_1) = (1-\epsilon F_1) n \\
W_2 &= F_2 W_1 (1-\epsilon) + (1-F_2) W_1 = (1-\epsilon F_2) W_1 = (1-\epsilon F_2)(1-\epsilon) F_1 n \\
&\vdots \\
W_t &= n \prod_{i=1}^{t} (1-\epsilon F_i) \leq n \prod_{i=1}^{t} e^{-\epsilon F_i} = n e^{-\epsilon \sum_{i=1}^{t} F_i}
\end{aligned}
$$

Randomized Weighted Majority Analysis (cont'd)

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

Randomized Weighted Majority Analysis (cont'd)

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

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

Randomized Weighted Majority Analysis (cont'd)

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

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

But best expert makes $\mathbf{m}$ mistakes

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

Randomized Weighted Majority Analysis (cont'd)

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

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

But best expert makes $\mathbf{m}$ mistakes

$$\implies W_t \ge (1-\epsilon)^m \implies \ln W_t \ge m \ln(1-\epsilon)$$

So $\mathbf{m \ln(1-\epsilon) \le \ln n - \epsilon E[M]}$

Randomized Weighted Majority Analysis (cont'd)

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

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

But best expert makes m mistakes

$$\implies W_t \ge (1-\epsilon)^m \implies \ln W_t \ge m \ln(1-\epsilon)$$

So $m \ln(1-\epsilon) \le \ln n - \epsilon E[M]$

$$\implies E[M] \le \frac{1}{\epsilon}(\ln n - m\ln(1-\epsilon)) \le (1+\epsilon)m + \frac{1}{\epsilon}\ln n$$

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