CS 170
Efficient Algorithms and Intractable Problems

Lecture 25
Online Algorithms 1

Nika Haghtalab and John Wright

EECS, UC Berkeley

Announcements

This week is the last week of class
→ Last week with discussion sections
→ Last required homework is out this week, we will have an optional homework next week

Final exam is on Monday May ~~3~~ 12, at 11:30am

Please fill out the course eval form!

Recall: Primality Test

Primality Testing: Given a number N, is it a prime number?

Fermat's Little Theorem

If p is a prime, then for all $x = 1, \dots, p-1$ we have that $x^{p-1} \equiv 1 \pmod{p}$

This suggests that we might be able to deduce whether N is a prime by looking at whether $x^{p-1} \not\equiv 1 \text{ (mod N)}$ for some choice of x. Let's choose x at random!

Fermat's Primality Test

Choose x uniformly at random from all $x = 1, \dots, N-1$.
Return "prime" if $x^{N-1} \equiv 1 \pmod{N}$, otherwise return "composite"

Recall: Composite N and Carmichael numbers

Let's say input was composite number $N = 9$. All arithmetic here is mod 9.

$1^8 \equiv 1$

$2^8 \equiv 4 \not\equiv 1$

$3^8 \equiv 0 \not\equiv 1$

$4^8 \equiv 7 \not\equiv 1$

$5^8 \equiv 7 \not\equiv 1$

$6^8 \equiv 0 \not\equiv 1$

$7^8 \equiv 4 \not\equiv 1$

$8^8 \equiv 1$

Out of 8 choices for a random $x \in \{1, \ldots, 8\}$, only 2 of them would lead Fermat's test to erroneously state that 9 is a prime! Fermat's test would have been correct with prob 0.75!

There are rare exceptions: There are composite numbers N for which $x^{N-1} \equiv 1 \pmod{N}$ for many xs.

Carmichael numbers:
Composite number N for which $x^{N-1} \equiv 1 \pmod{N}$ for all x that's coprime with N.

Correctness of the Primality Test

$N = \text{prime} \Rightarrow \text{output} = \text{prime} \checkmark$

Theorem: Assume that N is a composite, but not Carmichael number. Then with prob $> 1/2$ Fermat's outputs "composite". i.e.

$x^{N-1} \not\equiv 1 \ (mod\ N)$ for at least half of $x = 1, \dots, N-1 \rightarrow \frac{1}{2}$ are good.

1) N is not Carmichael $\Rightarrow \exists$ $\underline{\underline{a}}$ (good) coprime with N s.t. $a^{N-1} \not\equiv 1 \pmod N$

a coprime with $N \Rightarrow \exists\, \bar{a}^{1}$, s.t. $a \cdot \bar{a}^{1} \equiv 1 \pmod N$

2) Take any "bad" b_i (means $b_i^{N-1} \equiv 1 \pmod N$) $\Rightarrow \exists$ a good g_i ($g_i^{N-1} \not\equiv 1 \mod N$)

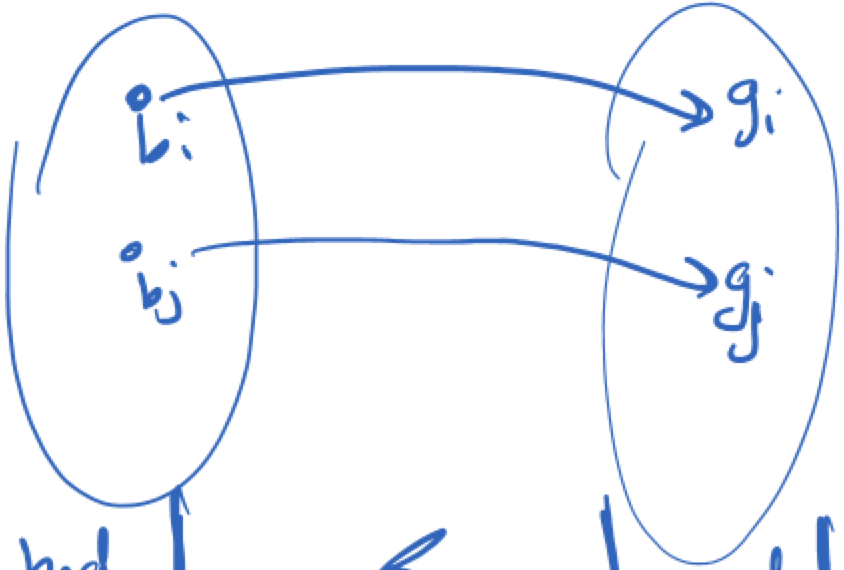
$(g_i)^{N-1} \equiv (a b_i)^{N-1} \equiv a^{N-1} \underbrace{\cancel{b_i^{N-1}}}_{1} \equiv a^{N-1} \not\equiv 1 \mod N.$

3) The mapping from b_i to g_i is one-to-one: If $b_i \neq b_j \Rightarrow g_i \neq g_j$

why? assume not $\underbrace{g_i}_{ab_i} \equiv \underbrace{g_j}_{ab_j} \xrightarrow{\times \bar{a}^1} \underbrace{\bar{a}^1 a}_{1} b_i \equiv \underbrace{\bar{a}^1 \cdot a}_{1} b_j \Rightarrow b_i \equiv b_j$

Correctness of the Primality Test (cont.)

We proved that for every **bad** b_i (for which $b_i^{N-1} \equiv 1\ (mod\ N)$) there is a distinct **good** $g_i = b_i a$ (for which $g_i^{N-1} \not\equiv 1\ (mod\ N)$)



$|\text{bad}| \leq |\text{good}|$

$b_i^{N-1} \equiv 1 \mod N$ $\qquad$ $g_i^{N-1} \not\equiv 1 \mod N$

$\Rightarrow |\text{good numbers}| = \left|\{x \mid x^{N-1} \not\equiv 1 \mod N\}_{<N}\right| \geq \frac{1}{2}$ of all $\{x<N\}$

$\uparrow$ good x

$\Downarrow$ Fermat's test w.p. $\geq \frac{1}{2}$ chose a good $x \Rightarrow$ composite N.

Primality Testing through the ages

200 BC: Eratosthenes (Greek polymath) described the *prime number sieve* for finding all the prime numbers up to a certain value.

1976: Miller and and Rabin came up with a randomized algorithm (similar to what we discussed but one more idea to deal with Carmichael numbers)

1977 …. 2002: Other randomized algorithms

2002: Agrawal, Kayal, and Saxena gave a polynomial time *deterministic* algorithm for primality testing (de-randomizing one of their earlier algorithms from 1999)

Online Algorithms

Online Algorithms

So far, we studied algorithmic problems where,

- Input given in one whole
- We generate output in one whole

But for some algorithmic problems, we are faced with

- Input that is given to us piece-by-piece
- Making irrevocable decisions: can’t wait to see the entire input, or future input depends on past and current decisions.

These are called online algorithms

(as opposed to offline)

Our focus: Algorithms for “online learning” that play a big role in Alg design, ML, etc.

Stock Market Predictions

Every day:

→Need to decide to invest or not.

→ I ask for advice from *"experts":* websites, influencers, and my toddler

→Experts recommend invest or not invest

→Market's up/down become clear after

End of the year:

→ Want investment decisions as best as the best experts would have recommended.

Online Routing

Every day:

→I need to decide which route to take to campus.

→ Traffic is not a priori known

→ Only after I arrive on campus, I know how long my commute took me.

End of the year:

→ Want my commute time to be short, as short as the best historical route.

Learning from Experts: Problem Setting

- There are n “experts” that have advice and opinion about each day
- Expert = someone with an opinion (but not necessarily correct)
- We want to make out own decision as to what’s going to happen

	Wallstreet Journal	**Co-worker**	**Motely Fool**	**TikTok Astrologer**	**My decision**	**Real outcome**
Day 1	down	up	up	up	up	up
Day 2	down	up	up	down	up	down
Day 3	up	up	down	down	down	down
Day 4	up	down	down	up	up	up

- Basic question: Is there a strategy that allows us to do nearly as well as best of these experts in hindsight?

Formalism:

There are n “experts”, $i = 1, \ldots, n$ and T days $t = 1, \ldots, T$

On each day $t = 1, \ldots, T$

dist guess

- All experts i give me their *opinion* $o_i^{(t)}$ (binary, like Yes/No, or Up/Down)

Alg, history; guess

- I make my prediction $guess^{(t)}$
- Afterwards, I see the real outcome $real^{(t)}$, which can be worst-case 
  - →Happy if guessed correctly and sad if I made a mistake!

My goal:

$$\underbrace{\sum_{t=1}^{T} \mathbf{1}\left(guess^{(t)} \neq real^{(t)}\right)}_{\text{\# of mistakes my Alg makes}} \lesssim \min_i \underbrace{\sum_{t=1}^{T} \mathbf{1}\left(o_i^{(t)} \neq real^{(t)}\right)}_{\text{\# of mistakes the best expert}} \boxed{+ \text{ small}}$$

A Simpler Setting

What if at least one of these n experts is perfect (makes 0 mistakes!) We just don't know which ones are perfect a priori.

What's an algorithm that is guaranteed to make a small number of mistakes?

Idea: Never follow an expert that's already made a mistake.

Attempt 1: Follow $1st$ expert's advice until they make a mistake … then follow the advice of the next expert who hasn't made a mistake yet, and repeat.
How well does this do?

Can make N mistakes,
but no more.

Halving Algorithm

Attempt 1: Every time Alg makes a mistake, we rule out 1 expert.

Atempt 2: Every time Alg makes a mistake, we rule out **many experts!**

How?

→ Follow the majority vote of the active experts (those with 0 mistakes so far)

Halving Algorithm

Let $E_1 = [n]$ //all experts are active

For $t = 1, \dots, T$

- $guess^{(t)} \leftarrow Yes$ iff at least half of the experts in E_t guess Yes
- $E_{t+1} \leftarrow \left\{ i \in E_t \mid o_i^{(t)} = real^{(t)} \right\}$ //Remove experts who were wrong

Example of Halving Algorithm

expert

	1	2	3	4	5	6	7	**My decision**	**Real Outcome**
t=1 Included in set E_1?	✓	✓	✓	✓	✓	✓	✓		
Opinions on day $t = 1$	Y	Y	N	Y	N	Y	N	Y	N
t=2 Included in set E_2?			✓		✓		✓		
Opinions on day $t = 2$			Y		Y		N	Y	Y
Included in set E_3?			✓		✓				
Opinions on day $t = 3$			N		N			N	N
Included in set E_4?			✓		✓				

Theorem: Bound on # Mistakes of Halving

When there is a perfect expert, Halving makes at most $\leq \boldsymbol{log_2(n)}$ mistakes

Proof: If we (W) make a mistake at time t, majority of E_t were wrong → $|E_{t+1}| \leq \frac{1}{2}|E_t|$. After $\log_2(n)$ mistakes, only one expert is left in the set.

Can we do better?

rule: $\exists$ a perfect expert. $\boxed{\checkmark}$

Theorem:

randomized: $\geq \frac{1}{2}\log_2(N)$

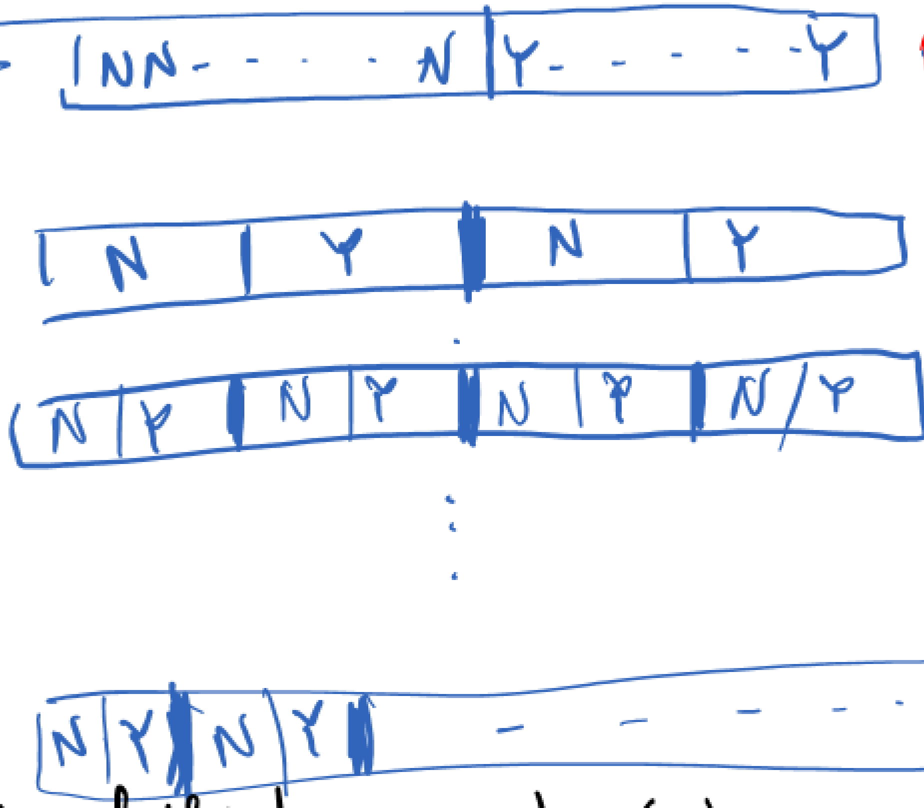
In the worst-case, any deterministic algorithm makes $\boldsymbol{log_2(n)}$ mistakes

Adversary: real$^{(t)}$

whatever guess$^{(t)}$ is

I claim $\boxed{\text{real}^{(t)} \neq \text{guess}}$

AlG guess$^{(t)}$



# of mistakes of AlG $\geq$ depth of this tree. $= \log(n)$

What if no perfect expert?

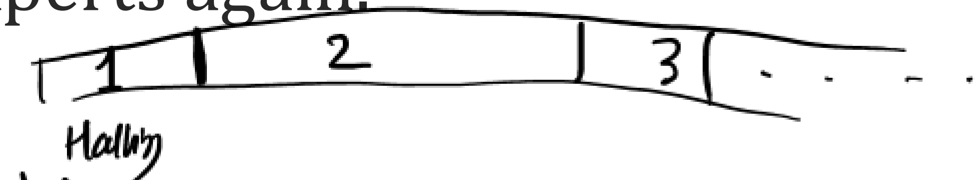
Halving completely rules an expert after their first mistake.

→No perfect expert? Don't rule out someone after their first mistake.

Suppose we know that the best expert makes M mistakes

→**Attempt 1:** Run Halving M times back to back. After all experts are thrown away, restart Halving with all experts again.

→How many mistakes does Alg make?



- In each phase. Alg can make $\leq \log(n)+1$ mistake

# of phases $\leq$ M phases that finish

+1 phase that might never end.

$$\leq (M+1)(\log(n)+1)$$

Can we do better?

$O(\log(n) \cdot M) \Rightarrow O(M + \log(n))$

Halving Algorithm:

- A mistake disqualifies an expert and we took the majority of the remaining experts.

Weighted Majority Algorithm:

- A mistake **lowers the weight** of an expert. (e.g., divide by 2)
- Predict with the **weighted** majority of the experts.

	1	2	3	4	5	6	7	**My decision**	**Real Outcome**
Weights at $t = 1$	1	1	1	1	1	1	1		
Opinions on day $t = 1$	Y	Y	N	Y	N	Y	N	Y	N
Included in set E_2?	½	½	1	½	1	½	1		
Opinions on day $t = 2$	N	N	Y	N	Y	N	N	N	Y
Included in set E_3?	¼	¼	1	¼	1	¼	½		

Weighted Majority Algorithm

Weighted Majority Algorithm is run using parameter $0 < \epsilon < 1$

Every time an expert makes a mistake, its weight is multiplied by $(1-\epsilon)$

(Deterministic) Weighted Majority with parameter ϵ

Initialize weights $w_i^{(1)} = 1$ for all $i \in [n]$.

For $t = 1, \ldots T$

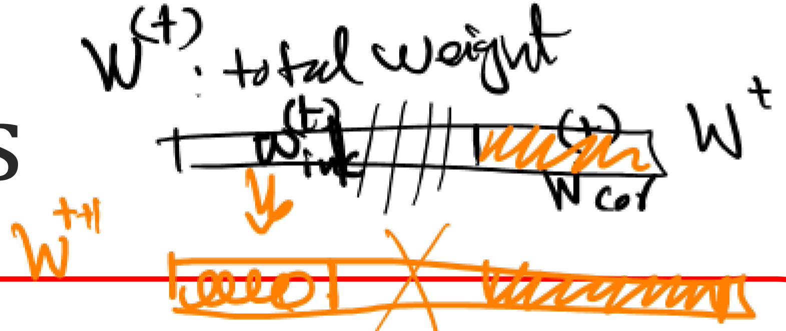
Take the weighted majority of the experts:

$$guess^{(t)} = \mathrm{argmax}_y \sum_{i\in[n]} w_i^{(t)}\, \mathbf{1}(o_i^{(t)} = y)$$

For $i = 1, \ldots, n$

If $o_i^{(t)} \neq real^{(t)}$ **then** $w_i^{(t+1)} \leftarrow w_i^{(t)}(1-\epsilon)$, else $w_i^{(t+1)} \leftarrow w_i^{(t)}$.

Weighted Majority Guarantees



Discuss

Assume Weighted Majority with $\epsilon = 0.5$ made a mistake on round t, what it the total weight of experts at time $t+1$ compared to the total weight of experts at time t?

a) $W^{(t+1)} \leq W^{(t)}/2$

b) $W^{(t+1)} \leq 3W^{(t)}/4$

c) $W^{(t+1)} = n/2$

d) $W^{(t+1)} \leq W^{(t)}/4$

1) $W_{inc}^{(t)} \geq \frac{1}{2} W^{(t)}$

2) $W^{t+1} = W^{t} - \frac{1}{2} W_{inc}^{(t)} \leq W^{t} - \frac{1}{2} \cdot \frac{1}{2} W^{t} \leq \frac{3}{4} W^{t}$

Assuming that expert i makes m_i mistakes, what is the weight of expert i when the algorithm quits?

a) $w_i^{(T+1)} = \left(\frac{1}{2}\right)^{m_i}$

b) $w_i^{(T+1)} = 1$

c) $w_i^{(T+1)} \geq \left(\frac{3}{4}\right)^{m_i}$

1 .... $\frac{1}{2}$ .... $\frac{1}{4}$ .... $\left(\frac{1}{2}\right)^{m_i}$

(t) .... 1st mistake ---- 2nd mist ---- m_ith

Proof of Weighted Majority Algorithm

Theorem: Guarantees of Weighted Majority $\epsilon = 0.5$

For M: Algorithms # mistakes and OPT: best expert's # mistakes, the (Deterministic) weighted majority algorithm with $\epsilon = 0.5$ gets

$$M \le 2.4(\log_2(\cancel{n}\ N) + OPT).$$

Proof: Let i^* be the optimal expert (OPT # mistakes)

$$\Rightarrow\quad w_{i^*}^{T+1} = \left(\tfrac{1}{2}\right)^{OPT} \text{ (prev slide)} \Longrightarrow W^{T+1} \geq \left(\tfrac{1}{2}\right)^{OPT} \quad (1)$$

(2) Every time Alg errs $W \leftarrow \tfrac{3}{4} W \Longrightarrow$ Alg makes M mistake $W^{T+1} \le \left(\tfrac{3}{4}\right)^{M} \cdot W^{1}$ (2)

(3) $W^1 = n$

(1), (2), (3)

$$\left(\tfrac{1}{2}\right)^{OPT} \le \left(\tfrac{3}{4}\right)^{M} \cdot N \Longrightarrow N \cdot 2^{OPT} \geq \left(\tfrac{4}{3}\right)^{M} \Rightarrow M \le 2.4\left(\log_2(n) + OPT\right)$$

$4\ \log(N) + OPT$