

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

**May 12, an earlier version of the slides had the wrong exam date**

Final exam is on Monday ~~May 8~~, 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  $x^{N-1} \not\equiv 1 \pmod{N}$  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  $x \in \{1, \dots, 8\}$ , **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  $x$ s.

## **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

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

$$x^{N-1} \not\equiv 1 \pmod{N} \text{ for at least half of } x = 1, \dots, N-1$$

---

# Correctness of the Primality Test (cont.)

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

# 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 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

Index ▲ 1.56 ▼ 0.78





# 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 our own decision as to what’s going to happen

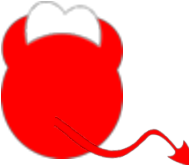
	Wallstreet Journal	Co-worker	Motely Fool	TikTok  Astrologer	My decision	Real outcome
Day 1	down	up	up	up	up	up

- 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”,  $i = 1, \dots, n$  and  $T$  days  $t = 1, \dots, T$

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

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

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

# 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?**

# Halving Algorithm

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

**Attempt 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**
- $E_{t+1} \leftarrow \{i \in E_t \mid o_i^{(t)} = real^{(t)}\}$  //Remove experts who were wrong

# Example of Halving Algorithm

	1	2	3	4	5	6	7	My decision	Real Outcome
Included in set $E_1$ ?	✓	✓	✓	✓	✓	✓	✓		

## Theorem: Bound on # Mistakes of Halving

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

# Can we do better?

## **Theorem:**

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



# 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?



# 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 $\epsilon$

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

For  $t = 1, \dots, T$

Take the weighted majority of the experts:

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

For  $i = 1, \dots, n$

**If**  $o_i^{(t)} \neq \text{real}^{(t)}$  **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 is 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)  $W^{(t+1)} \leq W^{(t)} / 4$

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

# 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

$$M \leq 2.4(\log_2(n) + OPT).$$

# How much do we regret?

*Alg's # mistakes*  $\leq 2.4(\log_2(|H|) + OPT)$  is good if  $OPT$  is small.

→ If best expert is wrong 5% of the time, we are wrong 12% of the time

→ If best expert is wrong 25% of the time, we are wrong half the time!

It would have been nice, if instead *Alg's # mistakes*  $- OPT \leq \text{small}$

→ Ideally, smaller than  $o(T)$ .

→ On average over  $T$  timesteps, we do nearly as well as the best expert.

Idea: Smoothly transition between predicting Yes or No based on the weights.

→ Weighted majority: 49% Yes, 51% No, we predict No

Randomized Weighted majority:

→ If 49% Yes, 51% No, we predict Yes with 0.49 prob and No with 0.51 prob.

→ We can also use less aggressive  $\epsilon$ .

# Randomized Weighted Majority

**Randomized** Weighted Majority Algorithm with parameter  $0 < \epsilon < 1$   
Every time an expert makes a mistake, its weight is multiplied by  $(1 - \epsilon)$

## Randomized Weighted Majority with parameter $\epsilon$

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

For  $t = 1, \dots, T$

Guess with probability proportional to the weighted majority:

$$\text{guess}^{(t)} \leftarrow y \text{ with prob. } \frac{1}{W^{(t)}} \sum_{i \in [n]} w_i^{(t)} \mathbf{1}(o_i^{(t)} = y)$$

For  $i = 1, \dots, n$

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

# Randomized Weighted Majority

## Theorem: Guarantees of Weighted Majority $\epsilon$

For  $M$ : Algorithms # mistakes and  $OPT$ : best expert's # mistakes, the **randomized weighted majority** algorithm with  $\epsilon$  gets

$$\mathbb{E}[M] \leq (1 + \epsilon)OPT + \frac{1}{\epsilon} \log_2(n).$$

For  $\epsilon = \sqrt{\frac{\log_2(n)}{OPT}}$ , get  $\mathbb{E}[M] \leq OPT + 2\sqrt{T \log_2(n)}$ .



# Beyond Binary Guesses and Outcomes

We can extend this to non-binary general outcomes and predictions

We want to take one of  $n$  actions, each one is like an “expert” E.g., each s-t path is one action/expert.

→ Each action  $i$  has some **cost** at time  $t$ , called  $c_i^{(t)} \in [0,1]$  E.g., The traffic of the  $i^{th}$  s-t path at time  $t$ .

→ Alg plays action  $i_t$  at time  $t$ , perhaps randomly

→ We see **cost** of all actions after we take an action

We want the total cost of the algorithm not to be much larger than the cost of the best action, in hindsight.

→ Want small regret

$$\text{REGRET} := \underbrace{\sum_{t=1}^T c_{i_t}^{(t)}}_{\text{Total cost of Alg's choices}} - \underbrace{\min_{i^*} \sum_{t=1}^T c_{i^*}^{(t)}}_{\text{Total cost of the best action}} \leq \textit{small}$$

# Multiplicative Weight Update (MWU) Algorithm!

## Multiplicative Weight Update with parameter $\epsilon$

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

For  $t = 1, \dots, T$

Play action  $i$  with probability  $\frac{w_i^{(t)}}{W^{(t)}}$

Observe costs  $c_i^{(t)}$  for all  $i = 1, \dots, n$

For  $i = 1, \dots, n$ ,

let  $w_i^{(t+1)} \leftarrow w_i^{(t)} (1 - \epsilon c_i^t)$

**Theorem:** For an appropriate choice of  $\epsilon = \sqrt{\log_2(n) / T}$ , the MWU Algorithm has

$$\mathbb{E}[\text{Regret}] \leq O\left(\sqrt{T \log_2(n)}\right).$$

# Next Lecture

Will be the last one!

We will see how useful online algorithms are!

And we will recap what we learned in this semester!