CPSC 420: Lecture 26
The Experts Problem and Multiplicative Weights

Joseph Poremba

March 22, 2022

Topics/Learning Objectives

Understand the Experts problem

- The model, objective, etc.

Understand the main algorithm

- Be able to run it, understand the analysis

(Hopefully) See some of the bigger picture

- The Multiplicative Weights algorithm
- Applications (Machine learning, linear programming)
- (but not understand in detail)

No supplementary reading, but sources for the brave:

- S. Arora, E. Hazan, and S. Kale, The Multiplicative Weights Update Method: A Meta-Algorithm and Applications
- N. Garg and J. Knemann, Faster and Simpler Algorithms for Multicommodity Flow and Other Fractional Packing Problems

Part I

The Experts Problem

Predicting Stocks

Suppose there is some stock on the market.

- Every day, the stock goes $\uparrow$ or $\downarrow$, we want to predict which
- The true stock performance is revealed each day after our prediction
- Minimize mistakes

Day	1	2	3	4	5
ALG	$\uparrow$	$\uparrow$	$\downarrow$		
Stock	$\uparrow$	$\downarrow$	$\uparrow$		
m_{ALG}	0	1	2		

Stock behaviour?

Adversarial

The stock performance is adversarial.

- Not random.
- Algorithm must do well in all cases.
- Could be designed to be evil and thwart our algorithm specifically.

Problem?

Not really predictable. We could be wrong every time! Doesn't seem fair...

Adding Experts

We get access to n "experts", who make their prediction first.

Day	1	2	3	4	5
e_1	↑	↓			
e_2	↓	↓			
e_3	↓	↑			
ALG	↓	↓			
Stock	↑	↓			
m_{ALG}	1	1			

Adding Experts

The expert predictions are also adversarial!

- They might not know what they are doing.
- Could also hate us, conspire against us, etc.

We only need to do well *relative to the experts*

- To make us do poorly, the input needs to make the experts do poorly

Goal

Don't make many more mistakes than the best expert.

Example: Naive Approach

Strategy 1: Follow the majority prediction

Day	1	2	3	4	5
e_1	↑	↑			
e_2	↓	↑			
e_3	↓	↓			
ALG	↓	↑			
Stock	↑	↓			
m^*	0	1			
m_{ALG}	1	2			

m^* vs. m_{ALG} ?

Example: Thwarting the Naive Approach

Strategy 1: Follow the majority prediction

Day	1	2	3	4	5
e_1	$\uparrow$	$\uparrow$	$\uparrow$		
e_2	$\downarrow$	$\downarrow$	$\downarrow$		
e_3	$\downarrow$	$\downarrow$	$\downarrow$		
ALG	$\downarrow$	$\downarrow$	$\downarrow$		
Stock	$\uparrow$	$\uparrow$	$\uparrow$		
m^*	0	0	0	0	0
m_{ALG}	1	2	3	4	5

The Weighted Majority Algorithm

Have a trust weight $w_i(t)$ for expert i at the start of day t
Choose the decision ($\uparrow$ or $\downarrow$) with the majority of total trust.

- i.e. compare $\sum_{i \text{ voting } \uparrow} w_i(t)$ and $\sum_{i \text{ voting } \downarrow} w_i(t)$

If an expert is wrong, penalize their weight, so we pay less attention to them in the future.

Update Rule

Initially, $w_i(1) = 1$ for all i. For every day after,

$$w_i(t+1) = \begin{cases} w_i(t), & \text{if } i \text{ is correct on day } t \\ \frac{1}{2} w_i(t), & \text{if } i \text{ is wrong on day } t \end{cases}$$

Multiplicative penalty is key.

The Weighted Majority Algorithm

Day	1	2	3	4	5
e_1	↑	↑	↑		
w_1	1	1	1		
e_2	↓	↓	↓		
w_2	1	1/2	1/4		
e_3	↓	↓	↓		
w_3	1	1/2	1/4		
$W^{\uparrow}$	1	1	1		
$W^{\downarrow}$	2	1	1/2		
ALG	↓	↓	↑		
Stock	↑	↑	↓		
m^*	0	0	1		
m_{ALG}	1	2	3		

Let:

- $w_i(t)$ = weight of expert i at the start of day t
- $m_i(t)$ = total mistakes of expert i at the end of day t

Goal: Discern some relationships. We want to relate the number of mistakes of the algorithm to m_i.

First: Relate m_i to w_i. Observe that $w_i(t)$ is halved for every mistake i makes.

$t \to t+1$

$$w_i(t+1) = \underbrace{\frac{1}{2} \times \frac{1}{2} \times \cdots \times \frac{1}{2}}_{\boxed{m_i(t) \text{ times}}} \overbrace{\boxed{w_i(1)}}^{1}$$

$$= \left(\frac{1}{2}\right)^{m_i(t)}$$

Now can we relate w_i to the mistakes of the algorithm?

Analysis of the Weighted Majority Algorithm

Let $W(t) = \sum_i w_i(t)$ (the "potential", or "total trust")
How does the potential evolve over time?

$$W(t+1) = \underbrace{\left(\sum_{i \text{ correct}} w_i(t)\right)}_{W^+(t)} + \frac{1}{2}\underbrace{\left(\sum_{i \text{ wrong}} w_i(t)\right)}_{W^-(t)}$$

$$= W^+(t) + \frac{1}{2}W^-(t)$$

Also: $W(t) = W^+(t) + W^-(t)$. Therefore...

$$W(t+1) = W(t) - W^-(t) + \frac{1}{2}W^-(t)$$

$$= W(t) - \frac{1}{2}W^-(t)$$

$W^-(t) \geq \frac{1}{2}W(t)$
$-\frac{1}{2}W^-(t) \leq -\frac{1}{4}W(t)$

Last slide: $W(t+1) = W(t) - \frac{1}{2}W^-(t)$
If the algorithm guesses wrong, $W^-(t) \geq \frac{1}{2}W(t)$. Then,

$$W(t+1) \leq W(t) - \frac{1}{2}\left(\frac{1}{2}W(t)\right) = \frac{3}{4}W(t)$$

So every time the algorithm is wrong, the potential goes down by a factor of at least 1/4.

Analysis of the Weighted Majority Algorithm

Say we have finished day T, start of day $T + 1$.
Say our algorithm has made $m_{\text{ALG}}(T)$ mistakes.
The total trust is *at most*

$$W(T+1) \leq \underbrace{\frac{3}{4} \times \frac{3}{4} \times \cdots \times \frac{3}{4}}_{m_{ALG} \text{ times}} W(1) = \left(\frac{3}{4}\right)^{m_{\text{ALG}}(T)} \cdot n.$$

From before: $w_i(T+1) \geq \left(\frac{1}{2}\right)^{m_i(T)}$.
Now, an easy bound: $w_i(T+1) \leq W(T+1) \leq \left(\frac{3}{4}\right)^{m_{\text{ALG}}(T)} \cdot n$.

$$\left(\frac{1}{2}\right)^{m_i(T)} \leq \left(\frac{3}{4}\right)^{m_{\text{ALG}(T)}} n$$

Analysis of the Weighted Majority Algorithm

Just algebra from here.

$$\begin{aligned}
\left(\frac{1}{2}\right)^{m_i(T)} &\leq \left(\frac{3}{4}\right)^{m_{\mathrm{ALG}}(T)} n \\
m_i(T) \cdot \log \frac{1}{2} &\leq m_{\mathrm{ALG}}(T) \cdot \log \frac{3}{4} + \log n && \text{take logs} \\
m_{\mathrm{ALG}}(T) &\leq 2.41 m_i(T) + O(\log n) && \text{rearranging}
\end{aligned}$$

Holds for any expert i, including the best one.

Theorem

The weighted majority algorithm with multiplicative penalty $\frac{1}{2}$ makes at most $2.41 m^*(T) + O(\log n)$ mistakes in the worst case.

Our error: $2.41m^*(T) + O(\log n)$

Theorem

Every deterministic algorithm has some input that forces it to make $\geq 2m^*(T)$ mistakes.

We can get very close to this by changing the $\frac{1}{2}$ weight penalty to a different fraction.

Theorem

The weighted majority algorithm with multiplicative penalty $(1-\epsilon)$, where $0 < \epsilon \leq \frac{1}{2}$, makes at most

$\epsilon \to 0$

$$2(1+\epsilon)m^*(T) + \frac{2\log n}{\epsilon}$$

2

mistakes in the worst case.

We can do even better (on average) with a randomized variant.

- Select one expert to listen to randomly.
- Use weights, penalize as before.
- Select expert i with probability $w_i(t)/W(t)$.

Theorem

This randomized algorithm, with multiplicative penalty $(1-\epsilon)$, *where* $0 < \epsilon \leq \frac{1}{2}$, *makes at most*

$$(1+\epsilon)m^*(T) + \frac{2\log n}{\epsilon}$$

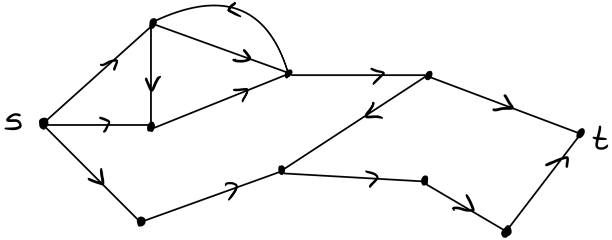
mistakes in $\boxed{\textit{expectation.}}$

Part II

The Bigger Picture

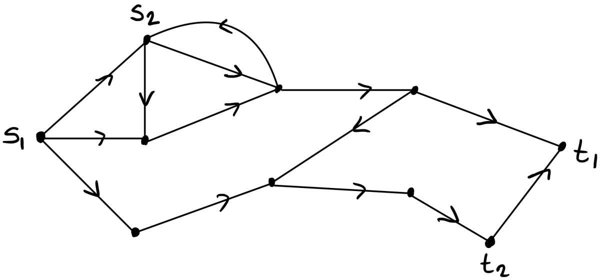
Flows??

Consider this completely unrelated problem.
Remember network flow?



Two-Commodity Flows??

Consider two-commodity flows:



Maximize k such that:

- there is an $s_1 t_1$ flow f_1 of size k
- there is an $s_2 t_2$ flow f_2 of size k
- $f_1(e) + f_2(e) \leq c(e) \quad \forall e \in E$ (shared bandwidth)

Multiplicative Weights Flow Algorithm

Can't just adapt Ford-Fulkerson!
Consider the following "algorithm".

$f_1, f_2 \leftarrow$ the zero-flow
$w(e) \leftarrow \delta$ for some semi-complicated δ, for every $e \in E$
while ...semi-complicated condition... **do**
  for $i \leftarrow 1, 2$ **do**
    $P \leftarrow$ a shortest (using lengths w) $s_i t_i$ path
    $\gamma \leftarrow \min_{e \in P} c(e)$
    $f_i \leftarrow f_i +$ push γ flow on P
    $w(e) \leftarrow \left(1 + \epsilon \frac{\gamma}{c(e)}\right) w(e)$ for $e \in P$
      ▷ Penalize selected edges
  end for
end while
Scale down f_1, f_2 until they are feasible

Multiplicative Weights Flow Algorithm

A special kind of approximation algorithm.

Theorem

This approach gives a polynomial-time approximation scheme (PTAS). For any fixed $\omega > 0$, we can tune the parameters δ, ϵ to make an algorithm that achieves:

- $(1 - \omega)$*-approximation*
- *running time* $O\left(\mathrm{poly}(|V|, |E|)\right)$ *(where ω is a hidden constant in the Big O)*

Multiplicative Weights

Experts, and this flow algorithm, are special cases of the *Multiplicative Weights "meta-algorithm"*.

- Use weights to guide decision making.
  - Experts: weight = how much we care about their opinion.
  - Flows: weight = how much we care about violating the edge's capacity.
- Penalize weights multiplicatively.

You can analyze both at the same time using one framework.

Applications of Multiplicative Weights

Online algorithms

- Online decision making (e.g. experts)
- Online convex optimization

Linear Programming

- Network flows
- Any "packing LP": constraints look like $Ax \leq b, x \geq 0$. $A \geq 0, b \geq 0$

Machine Learning

- Linear classification Winnow
- Boosting Ada Boost