LECTURE 25 (DECEMBER 4^{th})

ONLINE LEARNING

Consider the following online task:

- There are n experts that make binary predictions, e.g., weather the stock market goes up/down, whether the weather will be good or bad. The experts may be wrong.
- Each day all of them make a prediction about today's outcome
- We can see all of them and make our own prediction
- At the end of the day, the outcome is revealed and we know if we made the right prediction or not.

This process goes on indefinitely and our goal at any time is to not have too many more mistakes than the best expert in hindsight.

Here there is total uncertainty — no one knows what the outcome is going to be. Not even the experts! So, this is the best one can hope for, i.e., if an expert is correct 70% of the time in hindsight, we should also be correct almost 70% of the time.

This is an online decision making task because we do not know the future and still want to be competitive. This looks like an impossible task apriori.

Let's try some simple strategies in special cases –

1. Majority and halving

Suppose we know that the best expert makes no mistake but we don't their identity. Can we hope to make only a few mistakes?

Here's a strategy with $\log_2 n$ mistakes

- At each round we eliminate the experts that are wrong and among the remaining, we take the majority prediction

If we make a mistake, then the number of experts goes down by $1/2$ and after $\log_2 n$ mistakes, we have found the best expert and will not make any more mistakes

It is not hard to see that any deterministic strategy makes $\log_2 n$ mistakes, so one can't improve on it

Proof Day 1 First $\frac{n}{2}$ experts say 0

Last $\frac{n}{2}$ experts say 1

If we predict outcome 0, then the adversary chooses the real outcome to be 1 and vice-versa. So, we make one mistake.

Perfect expert is either in the interval $[1, \frac{n}{2}]$ or $[\frac{n}{2}+1, n]$

Day 2 In each interval $[1, \frac{n}{2}]$ and $[\frac{n}{2}+1, n]$ we make first half of experts say 0 and second half say 1.

Adversary chooses the real outcome to be opposite of what we predict, so we made another mistake.

Perfect expert is either in $[1, \frac{n}{4}]$ or $[\frac{n}{4}+1, \frac{n}{2}]$, $[\frac{n}{2}+1, \frac{3n}{4}]$ or $[\frac{3n}{4}+1, n]$

and so on!

Total mistakes $\geq \log_2 n$

2 Without a perfect expert

Suppose the best expert makes at most M mistakes
Here's a strategy that makes $\leq M(\log_2 n + 1)$ mistakes

Run the majority-halving algorithm above.
But if we discard all the experts, bring everyone back and start a new phase.

In each phase, each expert makes at least one mistake and we make at most $\log_2 n + 1$ mistakes. So, if the best expert only makes M mistakes, there are at most M phases and we make at most $M(\log_2 n + 1)$ mistakes

This is quite good as we are $\log_2 n+1$ competitive now but what is the best bound we can hope for

Cannot do better than the best expert who makes M mistakes. Cannot do better than $\log_2 n$ mistakes

Can we achieve a bound of $\approx M + \log_2 n$ mistakes?

Note that this is an additive guarantee.

Multiplicative Weights Algorithm — Class of algorithms

Throwing away an expert when they are wrong seems too drastic.

Suppose instead we give each expert a weight $w_i \geq 0$ and output the majority outcome according to the weights. If an expert is wrong we $\underbrace{\text{decrease its weight by 2}}_{\text{multiplicatively update the weights}}$. Each expert has weight 1 initially

$\boxed{\text{Theorem}}$ The above basic deterministic weighted majority algorithm makes $\leq 2.41\,(M + \log_2 n)$ mistakes

$\boxed{\text{Proof}}$ Potential function $\Phi_t = \sum_{i=1}^{n} w_i(t)$ ← weight of expert i at time t

$$\text{Initially} \quad \Phi_0 = n = \Phi_{init}$$

Each time we make a mistake, $\Phi_{new} \leq \frac{3}{4}\Phi_{old}$ since half the majority weight goes down by $\frac{1}{2}$

If we have made k mistakes,

$$\Phi_{final} \leq \left(\frac{3}{4}\right)^k \cdot \Phi_{init} = \left(\frac{3}{4}\right)^k \cdot n$$

If the best expert i^* has made M mistakes, then

$$w_{i^*} \geq \frac{1}{2^M} \implies \Phi_{final} \geq \frac{1}{2^M}$$

Thus, $$\left(\frac{3}{4}\right)^k n \geq \frac{1}{2^M} \implies \left(\frac{4}{3}\right)^k \leq n \cdot 2^M$$

$$\implies k \leq \frac{\log_2 n + M}{\log_2(4/3)} = 2.41\,(M + \log_2 n)$$

Thus, if the best expert is wrong 10% of the times, we are wrong about 24% of the times if $M \gg \log n$.

One can improve the factor of 2.41 arbitrarily close to 2 by decreasing the weights by a factor of $1-\varepsilon$ where ε is small.

Again no deterministic algorithm can make fewer than 2M mistakes

Suppose there are only two experts. One always says 0
Other always says 1

Fix any deterministic algorithm that makes prediction. The adversary can make sure that all the real outcomes are opposite of the prediction given by the algorithm. Thus, the algorithm makes a mistake everyday. On the other hand, the best expert must be right on $\gtrsim 50\%$ of the days.

Randomized Majority Algorithm

How about a randomized algorithm?

- Start with unit weights.
- At each time, pick a random expert where $\mathbb{P}[\text{expert } i \text{ picked}] = \frac{w_i}{\sum w_i}$

  Pick that expert's prediction as the answer
- For each expert that is wrong multiply its weight by $1-\varepsilon$

$\boxed{\text{Theorem}}$ Expected # of mistakes of the randomized weighted majority algorithm is at most

$$(1+\varepsilon) M + \frac{\ln n}{\varepsilon}$$

$\boxed{\text{Proof}}$ Potential $\Phi_t = \sum_{j=1}^{n} w_j(t)$ ← This only depends deterministically on the real outcome on all days and predictions of all the experts

Let F_t = fraction of total weight on the t^{th} day of experts who make a mistake on that day
= probability of making a mistake on the t^{th} day

$$\mathbb{E}[\text{Total \# mistakes}] = \sum_t F_t$$

On the t^{th} day, we claim that $\Phi_{new} = \Phi_{old}(1-\varepsilon F_t)$

To see this, let W_{wrong} = weight of experts who were wrong on day t
$W_{correct}$ = ______ correct ______

$$\text{Then, } F_t = \frac{W_{wrong}}{\underbrace{W_{wrong}}_{\substack{\text{weight goes}\\ \text{down by } (1-\varepsilon)}} + \underbrace{W_{correct}}_{\text{stays same}}} = \frac{W_{wrong}}{\Phi_{old}}$$

$$\begin{aligned}
\text{So, new potential } \Phi_{new} &= (1-\varepsilon) W_{wrong} + W_{correct} \\
&= (1-\varepsilon) W_{wrong} + (\Phi_{old} - W_{wrong}) \\
&= \Phi_{old} - \varepsilon \underbrace{W_{wrong}}_{\Phi_{old} \cdot F_t} \\
&= \Phi_{old}(1 - \varepsilon F_t)
\end{aligned}$$

$$\begin{aligned}
\text{Thus, } \Phi_{final} &= n \prod_t (1-\varepsilon F_t) \\
&\leq n \cdot e^{-\varepsilon \sum_t F_t}
\end{aligned}$$

$$\text{Also, } \Phi_{final} \geq (1-\varepsilon)^M, \text{ so}$$

$$n e^{-\varepsilon \sum_t F_t} \geq (1-\varepsilon)^M$$

$$\Longrightarrow \quad \varepsilon \sum_t F_t \leq M \ln \frac{1}{1-\varepsilon} + \ln n$$

$$\Longrightarrow \quad \underbrace{\sum_t F_t}_{\substack{\text{Expected \# of}\\ \text{mistakes}}} \leq M \underbrace{\frac{1}{\varepsilon} \ln \frac{1}{1-\varepsilon}}_{\leq (1+\varepsilon)} + \frac{\ln n}{\varepsilon}$$

Pretty sweet! We are theoretically close to the optimal bound

Suppose we want to look at error rates, i.e., what fraction of the time we make a mistake.

$$\frac{\text{Expected \# of mistakes}}{T} \leq (1+\varepsilon)\frac{M}{T} + \frac{\ln n}{\varepsilon T}$$

$$\text{average error rate} \leq \underbrace{\frac{M}{T}}_{\text{Error rate of the best expert } \leq 1} + \varepsilon \frac{M}{T} + \frac{\ln n}{\varepsilon T}$$

$$\leq \text{optimal error rate} + \varepsilon + \frac{\ln n}{\varepsilon T}$$

We can choose any ε we want, so let's choose one to minimize the above, $\varepsilon = \sqrt{\frac{\ln n}{T}}$

Then,

$$\text{our average error rate} \leq \text{optimal error rate} + \underbrace{2\sqrt{\frac{\ln n}{T}}}_{\text{Regret}}$$

The second term is called the regret and its vanishing (in the limit $T \to \infty$, it becomes zero)

Multiplicative weights algorithm can be generalized to more general scenarios where we have to pick one of the n outcomes and they have certain costs.

It was rediscovered multiple times in different fields and has applications in game theory, machine learning, optimization among others.

Feel free to explore them if you are interested, but we will have to end here!