CS49/249 (Randomized Algorithms), Spring 2021 : Lecture 22 + 23

Topic: Experts Problem : Weighted Majority, with and without randomness

Disclaimer: These notes have not gone through scrutiny and in all probability contain errors. Please discuss in Piazza/email errors to deeparnab@dartmouth.edu

---

- **The Experts Problem.** Suppose you want to predict the weather. Unfortunately, you have no idea of meteorology. But you have n friends who are *experts* (that is, they have a PhD; it doesn't mean they are infallible) who are willing to tell you their opinions. Your goal is to use their opinions to make a good prediction.

  Let's formalize. We denote $e_i(t) \in \{0,1\}$ to be expert i's prediction of whether it will rain or not on the tth day. The index i ranges from 1 to n. Before day t begins, you observe these $e_i(t)$'s and then you need to make a prediction $a(t) \in \{0,1\}$. Formally, you have to design an algorithm $\mathcal{A}$ which at step t takes input: $\{e_i(s) : 1 \le s \le t\}$ and $\{r(s), a(s)) : 1 \le s \le t-1\}$, and must output $a(t)$.

  Then the tth day unfolds, and you get to see $r(t) \in \{0,1\}$ whether it actually rained or not. If $a(t)$ doesn't match $r(t)$, say you pay a dollar – this is your *loss* $\ell(t)$ on day t. Thus, formally, $\ell(t) := 1 - a(t) \cdot r(t)$. We want to devise an algorithm $\mathcal{A}$ to make the total loss incurred, $\mathsf{loss} := \sum_{t=1}^{T} \ell(t)$, as small as possible. This is the experts problem.

  Remark: *This is a very important problem in machine learning where the business is making predictions based on past observations. The algorithm that we will see for this problem indeed has many applications and has been rediscovered in many different guises. In particular, although we are sticking to the loss being* 0 *or* 1*, one could imagine being it real valued, and indeed, in some situations, one can imagine the experts also being a continuous set of points. Once again, a whole course can probably be taught on this. We are just looking at couple of facets this week.*

- **Benchmark : Best Expert.** Note that this is not your usual computation problem with a fixed input and output. It has a flavor of a game that one is playing with nature, and in a sense nature (who is responsible for $r(t)$) gets to "play second". That is, our algorithm's decision has to be made under uncertainty of the outcome. Whenever this happens, it is not clear what "the best algorithm" even means. What is a good benchmark to compare one's algorithm against?

  Well, since our algorithm is deriving information from the experts, perhaps we should compare our losses with those of the n experts. In particular, at time t define $\ell_i(t)$ to be 1 if expert i makes a mistake at time i, and let $\mathsf{loss}_i := \sum_{t=1}^{T} \ell_i(t)$. But our experts can have differing qualities – which one should we choose our benchmark as? Well, it is clear that the "best expert" (the one with the smallest loss_i) is the most stringent among these benchmarks; let us see if we can design an algorithm which is as good as the best expert.

  Remark: *Note that in some case an algorithm can make more than the best expert. Imagine an algorithm which "knew" nature, that is, knew $r(t)$ before hand. Then it would just predict $a(t) = r(t)$ and would be better than the best expert unless the best expert also knew nature.*

- **Warm-up: There is a perfect expert.** Let's start with the case there is one "perfect expert". That is, there is some i^* with $\mathsf{loss}_{i^*} = 0$. If we knew who this perfect expert was, then our strategy is clear – predict whatever i^* is predicting. Can we quickly recognize this expert?

The answer is yes, and the algorithm is simple. It maintains a candidate set A of active experts; any expert not in A has made some mistake in the past and thus cannot be the perfect expert. Initially $A = [m]$, and we wish to whittle this set down fast. The MAJORITY algorithm is this: always predict what the majority of the experts in A are predicting (if $|A|$ is even, break ties arbitrarily). Every time our algorithm makes a mistake, that is, everytime loss increments by $+1$, we know that at least $|A|/2$ experts are whittled out. Thus, the maximum number of mistakes MAJORITY makes is $\log_2 n$.

Theorem 1. In the case of an perfect expert, the MAJORITY algorithm has regret $\leq \lg n$.

- **Weighted Majority.** However, perfect experts are mythical. What if we were only promised an expert who is correct 99% of the time? How would MAJORITY perform? Well, not too well as defined since the near-perfect expert could make a mistake on the first day along with the majority and be immediately whittled out. How should we fix this?

  Idea 1: Suppose we knew we were playing for T days. Keep taking MAJORITY till some expert makes $> 0.01T$ mistakes and then whittle them out. Two issues: one, it needs to know that some expert was as good as 99% (how would we know that?). More seriously, it's not a good algorithm – find an example where this algorithm fails badly.

  Idea 2: When an expert makes a mistake, instead of whittling them out, just decrease their "importance". More precisely, every expert has an importance or a weight $w_i(t)$. Initially, all the weights are 1; all experts are equal in our eyes. Each day, we go with the *weighted* majority. More precisely, for each choice we figure out the total *weight* of the experts predicting that value, and go with the larger one. Upon receiving the truth, that is, $r(t)$, we *update* the weights as follows: we penalize every expert which makes a mistake at time t by halving their weight. The full algorithm is described below.

```
1: procedure WEIGHTED MAJORITY:
2:     Maintain weights w_i() for each expert with w_i(1) = 1 for all i.
3:     for Days t = 1 to T do:
4:         Receive e_i(t) from each expert i.
5:         Let W_0 := sum_{i:e_i(t)=0} w_i(t) and W_1 := sum_{i:e_i(t)=1} w_i(t).
6:         Predict 0 if W_0 >= W_1, otherwise predict 1.
7:         Receive r(t) and incur loss l(t) = 1 if a(t) != r(t), and l(t) = 0 otherwise.
8:         for every i with e_i(t) != r(t) do:
9:             Set w_i(t+1) = w_i(t)/2
```

Theorem 2. If there exists an expert which makes at most k mistakes, then the WEIGHTED MAJORITY algorithm makes at most $2.41(k + \log_2 n)$ mistakes, where $2.41 \approx \frac{1}{\log_2(4/3)}$.

Corollary 1. If there is an expert who is correct at least 99% of the time, then if one plays for $T \gg \log n$ days, the WEIGHTED MAJORITY is correct $\geq 96\%$ of the time.

Proof. (of Theorem.) The analysis is similar in spirit to the analysis of the MAJORITY algorithm – we focus on the times t at which the algorithm makes a mistake. Consider such a time t when $a(t) \neq r(t)$.

By the algorithm's design we get to see at this t,

$$\sum_{i:e_i(t)\neq r(t)} w_i(t) \geq \sum_{i:e_i(t)=r(t)} w_i(t) \tag{1}$$

In plain English, the total weight of the experts that agreed with nature is less than the total weight of experts which disagreed.

Now note that for all experts in the LHS, their weight $w_i(t+1) = w_i(t)/2$ while for all experts in the right, $w_i(t+1) = w_i(t)$. Therefore, the *total weight* goes down by a ***multiplicative*** factor.

More precisely, let's define for any t, define the total weight as

$$Z(t) := \sum_{i=1}^{n} w_i(t)$$

Note that $Z(1) = n$ since $w_i(1) = 1$ for all $1 \leq i \leq n$. For any time t at which WEIGHTED MAJORITY makes a mistake, (1) implies $\sum_{i:e_i(t)=r(t)} w_i(t) \leq \frac{Z(t)}{2}$. Therefore, we get

$$\begin{aligned} Z(t+1) &= \sum_{i:e_i(t)\neq r(t)} w_i(t+1) + \sum_{i:e_i(t)=r(t)} w_i(t+1) \\ &= \sum_{i:e_i(t)\neq r(t)} \frac{w_i(t)}{2} + \sum_{i:e_i(t)=r(t)} w_i(t) \\ &= \frac{Z(t)}{2} + \frac{1}{2} \sum_{i:e_i(t)=r(t)} w_i(t) \\ &\leq \frac{3Z(t)}{4} \end{aligned}$$

The above was for the times when our algorithm makes a mistake. What about the times when it doesn't? Well, since the weights never *increase*, we have the trivial inequality $Z(t+1) \leq Z(t)$. Thus, if our algorithm makes ℓ mistakes (that is, $\ell = \mathsf{loss}$), after T rounds we get

$$Z(T) \leq (3/4)^{\ell} \cdot \underbrace{Z(1)}_{\text{which equals } n} \leq \left(\frac{3}{4}\right)^{\ell} \cdot n \tag{2}$$

So if our algorithm makes a lot of mistakes, the *potential* Z falls rapidly. Why is this at all useful in comparing with the best expert's loss? This is the second insight: if there is some expert making only k mistakes, then his weight at the end is *at least* $(1/2)^k$. Even if all the other experts are duds making tons and tons of mistakes, this expert forces

$$Z(T) \geq \frac{1}{2^k} \tag{3}$$

Now rest is arithmetic. From (2) and (3) we get

$$\frac{1}{2^k} \leq \left(\frac{3}{4}\right)^{\ell} \cdot n \quad \underbrace{\Rightarrow}_{\text{Taking log base 2, swapping signs, and rearranging}} \quad \ell \leq \frac{1}{\log_2(4/3)} \cdot (k + \log_2 n)$$

completing the proof. □

Remark: *It is not hard to show that no deterministic algorithm can get a better than factor* 2 *approximation. More precisely, for any strategy and for any* T*, there is a a collection of expert answers and "truths" such that the best expert makes* $\leq \delta$ *mistakes but the algorithm needs to make* 2δ *mistakes. In fact, this is true even with just* two *experts. Can you imagine such a scenario?*

- **Getting arbitrarily close with randomization.** Reiterating the theme that we have seen throughout the course, randomness allows us to get a better algorithm because it allows us to be more robust. This time more robust to the whims of nature. The flip is that we will have our result hold in expectation, and there will be a probability that it does poorly. In this lecture, I will stick only to the analysis of the expectation.

  Formally, the quantity $a(t)$ we predict is now random variable which 0 with a certain probability and 1 with the remainder. The *expected* number of mistakes made by the algorithm at time t is therefore the *probability* with which $a(t) \neq r(t)$. The total expected loss is the sum of these expectations (recall, linearity of expectation).

  What would be a candidate randomized algorithm? Note that in the WEIGHTED MAJORITY if 50.001% of the weight voted for 0 and 49.999% voted for 1, the algorithm went with 0. Even with this slim lead. The "fairer" randomized algorithm that suggests itself is that we should say 0 with probability 50.001% and 1 with the remainder probability. Turns out, this algorithm leads us arbitrarily close to the best expert!

  A more convenient way of looking at the RANDOMIZED WEIGHTED MAJORITY algorithm is by thinking of picking a random expert and going with their decision. That is, given the weights we pick an expert i proportional to $w_i(t)$, and then predict $a(t) = e_i(t)$. Observe that this is the same fair algorithm described above.

```
1: procedure RANDOMIZED WEIGHTED MAJORITY(η):▷ η ∈ (0, 1/2).
2:     Maintain weights w_i() for each expert with w_i(1) = 1 for all i.
3:     for Days t = 1 to T do:
4:         Select expert i with probability ∝ w_i(t).
5:         Predict a(t) = e_i(t).
6:         Receive r(t) and incur loss ℓ(t) = 1 if a(t) ≠ r(t), and ℓ(t) = 0 otherwise.
7:         for every i with e_i(t) ≠ r(t) do:
8:             Set w_i(t + 1) = w_i(t) · (1 − η) where η ∈ (0, 1/2).
```

Theorem 3. If there is an expert making at most k mistakes, then the *expected* number of mistakes made by the RANDOMIZED WEIGHTED MAJORITY(η) algorithm is

$$\mathbf{Exp}[\mathsf{loss}] \leq (1+\eta) \cdot k + \frac{\ln n}{\eta}$$

Proof. The proof idea is similar to the deterministic proof at a high level. As before, let $Z(t) := \sum_{i=1}^{n} w_i(t)$ be the sum of weights of all the experts at any time. Note we have (3) almost as is; since

there is an expert which makes at most k mistakes, their weight at the end would be $\geq (1-\eta)^k$. Since all weights are non-negative, we get

$$Z(T) \geq (1-\eta)^k \geq e^{-k(\eta+\eta^2)} \tag{4}$$

where have used our old friend $(1-x) \geq e^{-(x+x^2)}$ for all $x| \in (0, 0.5)$.

Let loss_t be the random variable indicating whether $a(t) \neq r(t)$, that is, whether we make a mistake on the tth day or not. Since $a(t) = e_i(t)$ if i is selected on day t, we get

$$\mathbf{Exp}[\mathsf{loss}_t] = \sum_{i:r(t)\neq e_i(t)} \mathbf{Pr}[i \text{ selected on day } t] = \frac{1}{Z(t)} \sum_{i:r(t)\neq e_i(t)} w_i(t) \tag{5}$$

Now we relate the total weights on day $t+1$ to those in day $t+1$ as

$$\begin{aligned} Z(t+1) &= \sum_{i:e_i(t)\neq r(t)} w_i(t+1) + \sum_{i:e_i(t)=r(t)} w_i(t+1) \\ &= (1-\eta) \sum_{i:e_i(t)\neq r(t)} w_i(t) + \sum_{i:e_i(t)=r(t)} w_i(t) \\ &= Z(t) - \eta \sum_{i:r(t)\neq e_i(t)} w_i(t) \underbrace{=}_{(5)} Z(t) - \eta Z(t)\, \mathbf{Exp}[\mathsf{loss}_t] \end{aligned}$$

This gives

$$Z(t+1) \leq Z(t) \cdot (1 - \eta\, \mathbf{Exp}[\mathsf{loss}_t]) \leq Z(t) \cdot e^{-(\eta\, \mathbf{Exp}[\mathsf{loss}_t])}$$

We have used $1 + x \leq e^x$ above. Why? Because it allows us to "telescope". We get

$$Z(T) \leq Z(1) \cdot e^{\left(-\eta \sum_{t=1}^{T} \mathbf{Exp}[\mathsf{loss}_t]\right)} = n \cdot e^{(-\eta \cdot \mathbf{Exp}[\mathsf{loss}])} \tag{6}$$

Taking natural logs on (6) and (4), gives

$$-(\eta + \eta^2)k \leq \ln n - \eta\, \mathbf{Exp}[\mathsf{loss}]$$

The theorem follows by "moving stuff around". □

- **The Notion of Regret.** The experts problem is an example of an *online learning* or *online optimization* problem, where we are designing algorithms which is competitive with unknowns in the future. In many such algorithms, one compares the performance of the algorithm and considers the *difference* with the best possible single strategy. In this case, this would be $\mathbf{Exp}[\mathsf{loss}]$, which is the performance of our algorithm, minus $\min_i \mathsf{loss}_i$, the best experts loss. This term is called the *regret* of the algorithm : what more it could have made if it were just simple-minded and stuck with the (unknown) best expert. $\mathbf{Reg}_T(\mathcal{A})$ is this difference after playing T rounds of this game. One desirable feature one shoots for is *sublinear* regret — $\mathbf{Reg}_T(\mathcal{A}) = o(T)$ for large T. That is, if we divide by T, then the "average difference" with the best strategy goes to 0 as we play long enough.

  Does Theorem 3 imply sublinear regret? Well we see that $\mathbf{Exp}[\mathsf{loss}] - k \leq \eta k + \frac{\ln n}{\eta}$. Note that k, the best expert, could be wrong a constant fraction of the times. Thus, ηk could be $\Omega(T)$. But if we chose $\eta = \frac{1}{\sqrt{T}}$, then we do see that both terms become $o(T)$. Indeed, choosing $\eta = \sqrt{\frac{\ln n}{T}}$ matches the two terms. So, if we knew that we were going to play for T rounds, then this indeed tells us what η to choose. This η is often called the "learning rate" or the "step size" in such algorithms.