COMS 4773: Using expert advice

Daniel Hsu

February 8, 2024

1 Using expert advice

Imagine you are conscripted to be the weatherperson for the morning news. This means that, every day, you must predict a $\{0,1\}$-valued outcome: say, rain or shine. This is now your day job, so you want to do well, i.e., make correct predictions as often as possible. You can determine whether or not your prediction was correct by the end of the day (having gazed out the window all day ...).

Of course, you know nothing about meteorology, so you have no idea how to make a good prediction on your own. But fortunately, you can consult your "expert" friends for advice at the start of each day. Each of these N experts makes their own $\{0,1\}$-valued prediction, and these predictions are available to you before you have to make your own prediction.

Because you are paranoid about doing well, you would like a prediction strategy under the assumption that Nature is adversarial. That is, you assume that Nature controls the weather as well as the experts, and also that Nature is out to make you look as foolish as possible. What strategy can you employ?

Without any further assumptions, there is nothing you can do. The adversarial Nature will just look at your prediction and just make the opposite happen. So you make a mistake every day.

2 Realizability

To make some progress, let's assume that Nature is constrained in how the outcomes and experts are controlled so that there is an expert who always makes correct predictions. This is called the *realizability assumption* for this problem.

A first algorithm to consider is CONSISTENT:

- Initialize V to be the list of all N experts, ordered however you like.
- Each day, predict the same as the first expert in V.
- At the end of each day, eliminate from V all experts who made incorrect predictions.

Proposition 1. *If the realizability assumption holds, then CONSISTENT makes at most $N-1$ mistakes.*

Proof. Every time you make a mistake, you eliminate at least one expert. But under the realizability assumption, there is an expert who will never be eliminated. You can therefore only eliminate at most $N-1$ experts. □

Not too shabby, but it turns out that you can do a lot better. Consider the following algorithm, called HALVING:

- This is the same as CONSISTENT, except that each day, you predict the same as the majority of the experts in V.

Proposition 2. *If the realizability assumption holds, then HALVING makes at most* $\log_2 N$ *mistakes.*

Proof. Every time you make a mistake, you eliminate at least half of the remaining experts. Just as in the proof of Proposition 1, under the realizability assumption, there is one expert who will never be eliminated. You can only halve the number of experts $\log_2 N$ times before you are left with just one expert. □

3 Non-realizability

Perhaps it is too much to hope for a perfect expert; Nature may not be so kind. But perhaps it is possible to do something even if all experts may make mistakes, as long as some expert does not make too many mistakes. Unfortunately, both CONSISTENT and HALVING may both get stuck without the realizability assumption, because they may end up eliminating all experts.

A simple fix is to simply "resurrect" all of the experts if they are ever all eliminated. That is, simply put all experts back into V. Call this algorithm HALVING++:

- Same as HALVING, except that if V is ever empty at the end of the day, then put all experts back into V.

Proposition 3. *For any t and K, if there is an expert who makes at most K mistakes after t days, then HALVING++ makes at most* $(K+1)\log_2 N$ *mistakes after t days.*

Proof. Suppose expert i makes at most K mistakes after t days. If all experts have been eliminated (and must be "resurrected"), then all experts—including expert i—must have made at least 1 mistake since the last "resurrection" (or the initial day). This means that you will only have to "resurrect" all experts at most K times after t days. Before/after "resurrections", you may make at most $\log_2 N$ mistakes since every mistake you incur corresponds to halving the number of experts in V (same argument as in the proof of Proposition 2). □

A more sophisticated algorithm, called WEIGHTED MAJORITY gets a "square-root" improvement over HALVING++. (It roughly improves from the product of K and $\log N$ to the sum.)

Theorem 1 (Littlestone and Warmuth, 1994)**.** *For any t and K, if there is an expert who makes at most K mistakes after t days, then WEIGHTED MAJORITY makes at most* $2.41(K + \log_2 N)$ *mistakes after t days.*

A fine-tuned version of WEIGHTED MAJORITY guarantees a bound of at most

$$2K + O\Big(\sqrt{K \log N} + \log N\Big)$$

mistakes after t days as long as there is an expert who makes at most K mistakes after t days. Thinking of K as a quantity that grows over time (i.e., $K \to \infty$), this bound is

$$(2 + o(1))K.$$

Before presenting WEIGHTED MAJORITY, consider the following question: could you hope to ensure a bound of $(1 + o(1))K$ mistakes? If so, then you would "asymptotically" (as $K \to \infty$) *perform nearly as well as the best expert*, no matter how many mistakes the best expert makes. You would never look (much) more foolish than an expert!

Sadly, the answer to this question is no. Consider the following scenario due to Cover (1965).

- There are $N = 2$ experts: Expert 0 and Expert 1.
  - Expert 0 always predicts 0.
  - Expert 1 always predicts 1.
- Nature always sets the outcome to be the opposite of your prediction.

In this scenario, you make a mistake every day. However, after t days, at least one of Expert 0 and Expert 1 has made at most $t/2$ mistakes. So for $t = 2K$, you will have made $2K$ mistakes even though there is an expert who makes at most K mistakes.

4 Weighted majority

Let us introduce some notation and formalize the problem we've been considering. For round $t = 1, 2, \ldots$:

- The learner (you) observes the predictions $b_{t,1}, \ldots, b_{t,N} \in \{-1, 1\}$ of the N experts.
- The learner must then make a prediction $a_t \in \{-1, 1\}$.
- Nature then reveals the outcome $y_t \in \{-1, 1\}$.

The total number of mistakes of the learner after t rounds is

$$\sum_{s=1}^{t} \mathbb{1}\{a_s \neq y_s\}.$$

(Here we have switched from $\{0, 1\}$ to $\{-1, 1\}$ for notational reasons that will become clear later.)

The basic version of WEIGHTED MAJORITY is the following algorithm, due to Littlestone and Warmuth (1994).

- Initially, set $w_{1,i} := 1$ for all $i \in [N] := \{1, 2, \ldots, N\}$. This is the "weight" of expert i.
- In round t:
  - Observe the experts' predictions $b_{t,1}, \ldots, b_{t,N} \in \{-1, 1\}$.
  - Predict $a_t = \operatorname{sign}\left(\sum_{i=1}^{N} w_{t,i} b_{t,i}\right) \in \{-1, 1\}$.
  - Observe the outcome $y_t \in \{-1, 1\}$.
  - Update experts' weights: for each $i \in [N]$,

$$w_{t+1,i} = \begin{cases} w_{t,i} & \text{if } y_t = b_{t,i}, \\ \frac{1}{2} w_{t,i} & \text{if } y_t \neq b_{t,i}. \end{cases}$$

Here, the sign($\cdot$) function maps negative numbers to -1 and positive numbers to 1. To break ties, let's say $\text{sign}(0) = -1$. (This choice won't really matter in the analysis of WEIGHTED MAJORITY.)

We can think of the weight of an expert as a measure of its reliability. We consider all experts to be equally reliable at the start. And experts who make a mistake have their reliability score reduced (halved).

We already gave the guarantee for WEIGHTED MAJORITY in Theorem 1. The idea behind the proof is similar to the analyses of CONSISTENT and HALVING. Every time the learner makes a mistake, we should be able to account for it in some way. With CONSISTENT and HALVING, we account for mistakes in terms of the number of experts eliminated; the difference between CONSISTENT and HALVING is how many experts are eliminated per mistake. In WEIGHTED MAJORITY, a mistake implies that half of the total weight of experts corresponds to experts' who were mistaken. So the proof of Theorem 1 will track the total weight of experts as it evolves over time. (We regard the total weight of experts as a "potential function".)

Proof of Theorem 1. For any round t, define

$$Z_t = \sum_{i=1}^{N} w_{t,i}$$

to be the total weight of experts at the start of round t, so $Z_1 = N$.

Suppose WEIGHTED MAJORITY makes a mistake in round t. Let α_t be the fraction (or proportion) of the total weight of experts that were mistaken. The weight of these experts are halved in the update; the weight of the remaining experts are unchanged. We know $\alpha_t \geq 1/2$ by the way WEIGHTED MAJORITY chooses its (mistaken) prediction. Therefore

$$\begin{aligned} Z_{t+1} &= (1-\alpha_t)Z_t + \frac{1}{2}\alpha_t Z_t \\ &= Z_t - \frac{1}{2}\alpha_t Z_t \\ &\leq Z_t - \frac{1}{4}Z_t \quad (\text{since } \alpha_t \geq 1/2) \\ &= \frac{3}{4}Z_t. \end{aligned}$$

Therefore, by induction, if WEIGHTED MAJORITY makes M mistakes after t rounds, then

$$Z_{t+1} \leq \left(\frac{3}{4}\right)^M Z_1 = \left(\frac{3}{4}\right)^M N.$$

The weight $w_{t+1,i}$ of expert i at the start of round $t+1$ is determined by the number of mistakes expert i has made after t rounds: if expert i has made M_i mistakes, then $w_{t+1,i} = 2^{-M_i}$. Therefore, if there is an expert who has made at most K mistakes after t rounds, then

$$Z_{t+1} \geq 2^{-K}.$$

We now have upper- and lower-bounds on Z_{t+1}, so we can combine the bounds

$$2^{-K} \leq Z_{t+1} \leq \left(\frac{3}{4}\right)^M N$$

and rearrange to get $M \leq (K + \log_2 N)/\log_2(4/3)$. □

We can "fine-tune" the WEIGHTED MAJORITY algorithm to get a better guarantee. Instead of multiplying the weight of mistaken experts by $1/2$, we change the update to multiplication by β, for some "hyperparameter" $\beta \in [0, 1]$. Of course, $\beta = 0$ corresponds to the HALVING algorithm: mistaken experts have their weight zeroed out. And $\beta = 1$ does not change the weights at all.

Working through the same analysis as in the proof of Theorem 1, but now using β in place of the $1/2$, we obtain the following.

Theorem 2 (Littlestone and Warmuth, 1994)**.** *For any t and K, if there is an expert who makes at most K mistakes after t days, then WEIGHTED MAJORITY (with hyperparameter $\beta \in (0, 1)$) makes at most*

$$\frac{K \log(1/\beta) + \log N}{\log(2/(1 + \beta))}. \tag{1}$$

mistakes after t days.

(The base of the logarithm clearly does not matter in (1), but to make it concrete, let's always assume log uses base e.)

Here is what the "mistake bound" in (1) looks like for different values of β.

β	Equation (1)
1/2	$\approx 2.41K + 3.48 \log N$
2/3	$\approx 2.22K + 5.48 \log N$
3/4	$\approx 2.15K + 7.49 \log N$
4/5	$\approx 2.12K + 9.49 \log N$

The limit of the pre-factor on K is 2 as $\beta \to 1$ (as can be checked by L'Hôpital's rule). But the pre-factor on $\log N$ diverges to $+\infty$ as $\beta \to 1$.

If we re-parameterize β using $\beta = 1 - \delta$ for $\delta \in (0, 1)$, then we can use a Taylor expansion to understand (1) as a function of δ, at least for δ close to zero. Concretely, there is an absolute constant $C > 0$ such that, if $\delta \in (0, 1/C)$, then

$$\frac{K \log(1/(1 - \delta)) + \log N}{\log(2/(2 - \delta))} \leq (2 + \delta)K + \frac{2}{\delta} \log N = 2K + \delta K + \frac{2 \log N}{\delta}.$$

To "balance" the last two terms, assuming $K > 2C^2 \log N$, we can set

$$\delta = \sqrt{\frac{2 \log N}{K}}$$

so that

$$\delta K + \frac{2 \log N}{\delta} = 2\sqrt{2K \log N}.$$

On the other hand, if $K < 2C^2 \log N$, then setting $\delta = 1/(C + 1)$ (for example) gives

$$\delta K + \frac{2 \log N}{\delta} \leq C' \log N$$

for some other absolute constant $C' > 0$ (that depends only on C). So either way, with this "fine-tuned" choice of δ (or, equivalently, of β), we ensure that WEIGHTED MAJORITY makes at most

$$2K + O\Big(\sqrt{K \log N} + \log N\Big)$$

mistakes after t rounds, as long as there is an expert who makes at most K mistakes after t rounds.

References

Thomas M Cover. Behavior of sequential predictors of binary sequences. *Transactions on Prague Conference on Information Theory Statistical Decision Functions, Random Processes*, pages 263–272, 1965.

Nick Littlestone and Manfred K Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.