CS 466/666: Algorithm Design and Analysis **University of Waterloo: Fall 2024**

Lecture 16

November 5, 2024

Instructor: Sepehr Assadi

Topics of this Lecture

1 The Experts Problem

In this lecture, we cover a new algorithmic technique, called the **Multiplicative Weight Update (MWU)** technique. You should really think of MWU as something like *dynamic programming* or *greedy* approaches in algorithm design; it is not a single algorithm but rather a general technique for designing algorithms for various problems and in various scenarios.

We start by looking at this technique for an abstract problem in online learning, sometimes called *the experts problem* or the problem of *learning from experts*. This will help motivating one of the uses of the MWU technique. In the next lecture then, we see how we can use MWU for solving some LP problems.

Problem 1. Suppose have a set of n "experts" who make predictions about a certain outcome each day (say, if is going to rain today or not). Abstractly, think of the *universe* of possible options for each day to be a set U. At the beginning of each day t:

1. Every expert $i \in [n]$ makes a **prediction** $p_i^{(t)} \in U$. Then, we, as the algorithm designer, receive all these predictions and have to come up with our **own prediction** $o^{(t)} \in U$ for that day.
2. After we made our prediction for that day, we get to see the **actual outcome** $a^{(t)} \in U$ of that day (say, whether or not it actually rained).

The goal in this problem is to minimize the number of **mistakes** we made, namely, the number of days that our own prediction was different from the actual outcome.

Notice that our only access to the "ground truth" of the universe is through these experts. This means that we cannot expect to minimize the number of mistakes in an *absolute* sense – if the experts are also making a lot of mistakes, there is no hope for us to not do the same (specifically, think of the universe U to be unbounded or even infinite, so it is not like we can just "guess" the right answer randomly at each

point). So, it makes sense to compare our number of mistakes with respect to the number of mistakes made by these experts.

It turns out that the *right* choice here is to compare ourselves with that of the *best expert* in hindsight; i.e., after the T days of the game are finished, we would like to say that our number of mistakes is not much larger that of the expert with the minimum mistakes in these T days – this allows us to minimize our "regret" in not having listened to this particular expert throughout. Thus, in this lecture, our goal is to minimize the number of mistakes we have to do compared to the number of mistakes of the best expert. Specifically, let m denote the number mistakes by our algorithm, and m_i denote the mistakes of the expert $i \in [n]$. Our goal is for m to be close to $m^\star = \min_{i \in [n]} m_i$.

1.1 Warm-Up 1: When there is an "ideal" expert

Let us first consider an easy case of this problem when $m^\star = 0$, i.e., there is an expert that does not make any mistake. Here, there is a very simple algorithm:

Algorithm 1. An algorithm for the expert problem when $m^\star = 0$.

1. Let N be the set of all experts initially.
2. Every day, return the *majority* predication of all experts in N as your own prediction (breaking the ties arbitrarily).
3. Upon seeing the actual outcome, remove any expert from N whose predication was wrong.

Lemma 1. *Algorithm 1 makes $\lceil \log n \rceil$ mistakes in total.*

Proof. Let $N^{(t)}$ denote the set of all experts still present at the beginning of a day $t \geqslant 1$. Suppose we make a mistake at day t; then, we know that at least half the experts in $N^{(t)}$ also made a mistake (because we went with the majority prediction that turns out to be wrong). Thus, we will have

$$|N^{(t+1)}| \leqslant \frac{1}{2} \cdot |N^{(t)}|.$$

In other words, for any mistake Algorithm 1 does, the number of remaining experts gets divided by half. At the same time, by our assumption that $m^\star = 0$, there is an expert that is never removed from N. Thus, $N^{(t)} \geqslant 1$ for all $t \geqslant 1$. This means that the total number of mistakes we can have is at most $\lceil \log n \rceil$ (as otherwise we will run out of experts which cannot happen). □

1.2 Warm-Up 2: A "Crude" Approximation

Suppose now we do not have the assumption that $m^\star = 0$, i.e., we are in the general case. Let us start with a very crude approximation algorithm, that is a direct generalization of Algorithm 1.

Algorithm 2. A direct generalization of Algorithm 1 to the case $m^\star > 0$.

1. Run Algorithm 1 and remove any expert with a mistake from the list of experts N as before.
2. Once N becomes entirely empty, bring back all the experts in N again and repeat from Line 1.

Lemma 2. *Algorithm 2 makes $m^\star \cdot \lceil \log n \rceil + \lceil \log n \rceil$ mistakes in total.*

Proof. Let $i^\star$ be the index of the best expert so $m_{i^\star} = m^\star$. Consider one *epoch* as the consecutive days when we start with N equal to all experts until it becomes completely empty and we restart. By the same exact

argument as in Lemma 1, during an epoch, the total number of mistakes we have is $\lceil \log n \rceil$. On the other hand, the total number of epochs is at most $m^\star$ because each time N becomes empty, it means $i^\star$ made a mistake, but the total mistakes of $i^\star$ is $m^\star$. Finally, after all the epochs, there can be at most another $\lceil \log n \rceil$ more mistakes (and this time N will not become empty) by Lemma 1 again. This proves the lemma. □

2 The Weighted Majority Algorithm

Let us now consider the first "real" algorithm for the problem.

It is clear that Algorithm 2 is quite "aggressive": it removes an expert based on just one mistake, and then brings them back at a later time without even taking into account the mistakes made by the algorithm throughout the process. So, we should consider a more "smooth" version of this algorithm, called the *weighted majority algorithm*: basically, we assign weights to the predictions of the experts and whichever expert that makes a mistake will get a lower weight – when it comes to predicate the outcome, we instead work with the weighted majority of the experts (hence, the name). Formally,

Algorithm 3. The weighted majority algorithm.

1. For every expert $i \in [n]$, maintain a weight $w_i^{(t)}$ for every day t. Initially, $w_i^{(t)} = 1$ for all $i \in [n]$.
2. Every day $t \geqslant 1$, return the weighted majority based on $w^{(t)}$ of the predications of all experts in N as your own prediction (breaking the ties arbitrarily).
3. Upon seeing the outcome, update the weight of any wrong expert $i \in [n]$ to be $w_i^{(t+1)} = (1-\eta) \cdot w_i^{(t)}$ for some parameter $\eta \in (0, 1)$ that we can choose later.

Lemma 3. *Algorithm 3 makes at most* $2 \cdot (1+\eta)\, m^\star + \dfrac{2 \ln n}{\eta}$ *mistakes in total for any* $\eta \in (0, 1)$.

Proof. Define the following *potential function* for every $t \geqslant 1$,

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

Consider any day $t \geqslant 1$ that we make a mistake in. We have,

$$\begin{aligned}
W^{(t+1)} &= \sum_{i:\ \text{correct}} w_i^{(t+1)} + \sum_{i:\ \text{wrong}} w_i^{(t+1)} && \text{(by the definition of } W^{(t+1)}\text{)} \\
&= \sum_{i:\ \text{correct}} w_i^{(t)} + \sum_{i:\ \text{wrong}} (1-\eta) \cdot w_i^{(t)} && \text{(by the definition of the update rule)} \\
&= W^{(t)} - \eta \cdot \sum_{i:\ \text{wrong}} w_i^{(t)} && \text{(by the definition of } W^{(t)}\text{)} \\
&\leqslant W^{(t)} - \frac{\eta}{2} \cdot W^{(t)} && \text{(because if we made a mistake, the weighted majority also made a mistake)} \\
&= (1 - \frac{\eta}{2}) \cdot W^{(t)}. && (1)
\end{aligned}$$

Let m denote the total number of mistakes we made till the last day T. Then,

$$W^{(T)} \leqslant (1 - \frac{\eta}{2})^m \cdot W^{(1)} = (1 - \frac{\eta}{2})^m \cdot n,$$

where we used Eq (1) repeatedly for the first inequality, and the fact that for all $i \in [n]$, $w_i^{(1)} = 1$ for the equality. This means that as long as we make "many" mistakes, the value of our potential function drops "a

lot" also. This is similar to the first part of the proof of Lemma 1 that argued that each mistake cuts the number of export—which can be think of the potential function in that lemma—by half.

On the other hand, we argue that the potential function cannot drop by a lot either. Let $i^\star$ be the index of the best expert so $m_{i^\star} = m^\star$. We have

$$w_{i^\star}^{(T)} = (1-\eta)^{m^\star},$$

because we only reduce the weight of $i^\star$ if it makes a mistake, and it can only make a mistake $m^\star$ times. But since $w_{i^\star}^{(T)} \leqslant W^{(T)}$, we have,

$$(1-\eta)^{m^\star} = w_{i^\star}^{(T)} \leqslant W^{(T)} \leqslant (1-\frac{\eta}{2})^m \cdot n.$$

By taking the natural log of both sides, we have,

$$m^\star \cdot \ln(1-\eta) \leqslant m \cdot \ln(1-\frac{\eta}{2}) + \ln n. \tag{2}$$

We now use the following two inequalities for $x \in (0, 1/2)$,

$$-\left(x + x^2\right) \leqslant \ln(1-x) \leqslant -x,$$

by the Taylor expansion of $-\ln(1-x) = x + \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{4} + \cdots$.

Applying these inequalities to Eq (2), we get

$$-m^\star \cdot \left(\eta + \eta^2\right) \leqslant -m \cdot \frac{\eta}{2} + \ln n,$$

which, by reorganizing terms, implies that

$$m \leqslant 2 \cdot (1+\eta) \cdot m^\star + \frac{2\ln n}{\eta},$$

as desired. □

Lemma 3 now says that we can get much closer the mistakes of the best expert, almost by a factor 2, at the cost of paying some extra additive factors of $\ln n$. Can we avoid this factor 2? The answer is *No*.

Proposition 4. *Any deterministic algorithm for the expert problem makes $\geqslant 2 \cdot m^\star$ mistakes on some inputs.*

Proof. Consider a setting that only consists of two experts, one of them always predicting 0 and the other, always predicting 1. Now, since the algorithm is deterministic, at any given day, we can make sure the output prediction of the algorithm is wrong. But, since there are two experts and with binary answers, at least one of the experts can only make an error half the days. So, our predictions can be wrong at least twice as much as $m^\star$ concluding the proof. □

Remark. In all the algorithms we analyzed so far, we could have actually only "penalize" the experts *only* on the days when our prediction ended up becoming wrong, not on every day (that the expert was wrong). It is easy to see that our analysis never took into account the mistakes by the experts on other days. But, that seems inefficient although Proposition 4 suggests that we could have not improved this much further. We show how to exploit this using *randomization* in the next algorithm.

3 The Multiplicative Weight Update Algorithm

Let us now present the final algorithm which is *randomized* and is often considered as 'the' multiplicative weight update (MWU) algorithm (even though, all other algorithms we saw also can be called MWU technically). We are going to use randomization in a very simple way to bypass the limit posed in Proposition 4 – basically, and for the most part, to break the ties.

Algorithm 4. The weighted majority algorithm.

1. For every expert $i \in [n]$, maintain a weight $w_i^{(t)}$ for every day t. Initially, $w_i^{(t)} = 1$ for all $i \in [n]$.
2. Every day $t \geqslant 1$, pick one expert $i \in [n]$ randomly such that the probability that expert $i \in [n]$ is chosen is proportional to its weight, i.e., is
$$\frac{w_i^{(t)}}{\sum_{j=1}^{n} w_j^{(t)}};$$
then return the prediction of this expert as your own prediction.
3. Upon seeing the outcome, update the weight of any wrong expert $i \in [n]$ to be $w_i^{(t+1)} = (1-\eta) \cdot w_i^{(t)}$ for some parameter $\eta \in (0,1)$ that we can choose later.

Notice that the only difference between Algorithm 4 and our previous Algorithm 3 is on how we output our own prediction – the update rule is exactly as before.

Lemma 5. *Algorithm 3 makes at most* $(1+\eta)\, m^\star + \dfrac{\ln n}{\eta}$ *mistakes* in expectation *for any* $\eta \in (0,1)$.

Proof. As in Lemma 3, define the following potential function for every $t \geqslant 1$,
$$W^{(t)} := \sum_{i=1}^{n} w_i^{(t)}.$$
Moreover, for every $t \geqslant 1$, define
$$B^{(t)} := \frac{1}{W^{(t)}} \cdot \sum_{i:\ \text{wrong}} w_i^{(t)}.$$
Notice that at any day $t \geqslant 1$,
$$\Pr\left(\text{output prediction is wrong}\right) = B^{(t)},$$
because we are picking the experts at day t according to their weight $w^{(t)}$. As such, if denote the number of mistakes in all the T days in our algorithm by m, we get that
$$\mathbb{E}\left[m\right] = \sum_{t=1}^{T} B^{(t)},$$
by linearity of expectation.

On the other hand, for *every* day $t \geqslant 1$ (and not necessarily the ones we make a mistake in),
$$\begin{aligned} W^{(t+1)} &= \sum_{i:\ \text{correct}} w_i^{(t+1)} + \sum_{i:\ \text{wrong}} w_i^{(t+1)} && \text{(by the definition of } W^{(t+1)}\text{)} \\ &= \sum_{i:\ \text{correct}} w_i^{(t)} + \sum_{i:\ \text{wrong}} (1-\eta) \cdot w_i^{(t)} && \text{(by the definition of the update rule)} \end{aligned}$$

$$
\begin{aligned}
&= W^{(t)} - \eta \cdot \sum_{i:\ \text{wrong}} w_i^{(t)} && \text{(by the definition of } W^{(t)}\text{)} \\
&\leqslant W^{(t)} - \eta \cdot B^{(t)} \cdot W^{(t)} && \text{(by the definition of } B^{(t)}\text{)} \\
&= (1 - \eta \cdot B^{(t)}) \cdot W^{(t)}.
\end{aligned}
$$

Let $i^\star$ be the index of the best expert so $m_{i^\star} = m^\star$. As before,

$$w_{i^\star}^{(T)} = (1-\eta)^{m^\star},$$

because we only reduce the weight of $i^\star$ if it makes a mistake, and it can only make a mistake $m^\star$ times. But since $w_{i^\star}^{(T)} \leqslant W^{(T)}$, we have,

$$(1-\eta)^{m^\star} = w_{i^\star}^{(T)} \leqslant W^{(T)} \leqslant \prod_{t=1}^{T} (1 - \eta \cdot B^{(t)}) \cdot n \leqslant \exp\left(-\sum_{t=1}^{T} \eta \cdot B^{(t)} + \ln n \right) = \exp\left(-\eta \cdot \mathbb{E}\left[m\right] + \ln n \right),$$

where we used the inequality $1 - x \leqslant e^{-x}$ for $x \in (0,1)$.

By taking the natural log of both sides, we have,

$$m^\star \cdot \ln(1-\eta) \leqslant -\eta \cdot \mathbb{E}\left[m\right] + \ln n.$$

And using the inequality $\ln(1-x) \geqslant -(x + x^2)$ (as shown in Lemma 3), we get,

$$-m^\star \cdot \left(\eta + \eta^2\right) \leqslant -\eta \cdot \mathbb{E}\left[m\right] + \ln n,$$

which, by reorganizing terms, implies that

$$\mathbb{E}\left[m\right] \leqslant (1+\eta) \cdot m^\star + \frac{\ln n}{\eta},$$

as desired. □

This brings us to a bound which is almost as good as the best expert, modulo a multiplicative loss which can be made arbitrarily small at a cost of paying an additive factor.

Now that we practiced several iterations of MWU-type algorithms and how to analyze them, we are ready for using this technique for designing algorithms for some classical problems in LPs. This will be the topic of the next lecture.

You can also learn (a lot) more about MWU, its origin, and various extensions and applications in the survey by Arora, Hazan, and Kale [AHK12].

References

[AHK12] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory Comput.*, 8(1):121–164, 2012. 6