Algorithms I May 27, 2025

Lecture 27: Decisions with Expert Advice

Notes by Ola Svensson[1]

This section is basically taken verbatim from the excellent lecture notes[2] by Anupam Gupta available here (together with a lot of other interesting information):

https://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15859-f11/www/

The Multiplicative Weights Update Algorithm is a simple yet powerful method that appears in many areas of computer science, including machine learning, optimization, game theory, and theoretical computer science. At its core, the algorithm maintains a set of weights over different options or experts and updates these weights multiplicatively based on their performance. This approach allows us to combine advice or predictions in a way that quickly adapts to changing environments and minimizes mistakes or losses. Because of its versatility and effectiveness, the multiplicative weights method has become a fundamental tool for designing algorithms with strong performance guarantees in a wide range of applications.

1 Prediction with Expert Advice

The following sequential game is played between an omniscient Adversary and an Aggregator who is advised by N experts. Special cases of this game include predicting if it will rain tomorrow, or if the stock market will go up or down.

Strategy 1 Prediction with Expert Advice

1: **for** $t = 1 \dots T$ **do**
2: Each expert $i \in [N]$ advises either YES or NO
3: Aggregator predicts either YES or NO
4: Adversary, with knowledge of the expert advice and Aggregator's prediction, decides the YES/NO outcome.
5: Aggregator observes the outcome and suffers if his prediction was incorrect.
6: **end for**

Naturally, Aggregator wants to make as few mistakes as possible. Since the experts may be unhelpful and the outcomes may be capricious, Aggregator can hope only for a relative performance guarantee. In particular, Aggregator hopes to do as well as the best single expert in hindsight[3]. In order to do so, Aggregator must track which experts are helpful. We will consider a few tracking strategies. Almost every other aspect of the game - that advise is aggregated into a single value, that this value is binary, and even that the game is sequential - is not relevant; we will generalize or eliminate these aspects.

If there is a perfect expert, then an obvious strategy is to dismiss experts who are not perfect. With the remaining experts, take a majority vote. Whenever the Aggregator makes a mistake, at least half of the remaining experts are dismissed, so Aggregator makes at most $\log N$ mistakes[4]. We can use the same strategy even when there isn't a perfect expert, if we restart after every expert has been eliminated. If the best expert has made M mistakes by time T, then Aggregator has restarted at most $M+1$ times, so it has made at most $(M+1) \cdot \log N$ mistakes. This bound is rather poor since it depends multiplicatively on M.

[1] **Disclaimer:** These notes were written as notes for the lecturer. They have not been peer-reviewed and may contain inconsistent notation, typos, and omit citations of relevant works.

[2] This section corresponds to Lecture 16 in the course, which was scribed by Shiva Kaul

[3] The excess number of mistakes is called (external) regret

[4] We will use log to denote the binary logarithm

2 Fewer Mistakes with Weighted Majority

We may obtain an additive mistake bound by softening our strategy: instead of dismissing experts who erred, discount their advice. This leads to the *Weighted Majority Algorithm* of Littlestone and Warmuth [2]. Assign each expert i a weight $w_i^{(1)}$ initialized to 1. Then for every t, predict YES/NO based on the weighted majority vote and halve the mistaken experts' weights after observing the outcome. The game strategy then becomes:

Strategy 2 Prediction with Weighted Majority

1: Initialize $\mathbf{w}^{(1)} = (w_1^{(1)}, \ldots, w_N^{(1)})$ to be a vector of 1's
2: **for** $t = 1 \ldots T$ **do**
3: Each expert $i \in [N]$ advises either YES or NO
4: Aggregator predicts either YES or NO based on a weighted majority vote using $\mathbf{w}^{(t)}$
5: Adversary, with knowledge of the expert advice and Aggregator's prediction, decides the YES/NO outcome.
6: Aggregator observes the outcome and for every mistaken expert i, set $w_i^{(t+1)} \leftarrow w_i^{(t)}/2$
7: **end for**

Claim 1 *For any sequence of outcomes, duration T, let M_{WM} be the number of mistakes that the Weighted Majority strategy makes, and M_i be the number of mistakes that expert i makes. Then*

$$M_{WM} \leq 2.41 \cdot (M_i + \log N)$$

Proof Let

$$\Phi^{(t)} = \sum_{i \in [N]} w_i^{(t)}$$

be a 'potential' function. Observe that:

- By definition, we have $\Phi^{(1)} = N$
- Also by definition, $\left(\frac{1}{2}\right)^{M_i} \leq \Phi^{(T+1)}$
- At any time τ when WM errs, at least half of the weight gets halved:

$$\Phi^{(\tau+1)} \leq \frac{3}{4}\Phi^{(\tau)}$$

This implies

$$\Phi^{(T+1)} \leq \left(\frac{3}{4}\right)^{M_{WM}} \Phi^{(1)}$$

Combining these facts yields

$$\left(\frac{1}{2}\right)^{M_i} \leq \left(\frac{3}{4}\right)^{M_{WM}} N$$

Taking the base-2 logarithm of both sides,

$$-M_i \leq \log N + \log\left(\frac{3}{4}\right) M_{WM}$$

so

$$M_{WM} \leq \underbrace{\frac{1}{\log(4/3)}}_{\approx 2.41}(M_i + \log N)$$

■

The leading constant is a consequence of our arbitrary choice to halve the weights. We can optimize ϵ in the update rule

$$w_i^{(t+1)} \leftarrow w_i^{(t)}/(1+\epsilon)$$

then using the WM strategy we may achieve

$$M_{WM} \leq 2(1+\epsilon)(M_i + O(\log N/\epsilon))$$

3 Changing the game: allowing for randomized strategies

In the exercises, you will show that there are instances for which weighted majority does twice as many mistakes as the best expert! This is undesirable. To overcome this limitation, we will allow the aggregator to play a random strategy instead of always making a deterministic prediction (that the adversary then uses to create bad instances). A side note is that this is often a general strategy: randomization is often very good to limit the effect of adversaries.

Allowing for randomized strategies leads to the following game with T days and N experts:

For $t = 1, \ldots, T$:

1. Each expert $i \in [N]$ gives some advice.
2. Allocator picks some distribution $\vec{p}^{(t)} = \left(p_1^{(t)}, \ldots, p_N^{(t)}\right)$ over the experts.
3. Adversary, with knowledge of the expert advice and $\vec{p}^{(t)}$, determines a cost vector $\vec{m}^{(t)} = (m_1^{(t)}, \ldots, m_N^{(t)}) \in [-1, 1]^N$.
4. Allocator observes the cost vector and suffers $\vec{p}^{(t)} \cdot \vec{m}^{(t)} = \sum_{i \in [N]} p_i^{(t)} m_i^{(t)}$.

Note that in the above game, we have abstracted away the given advice and in fact Step 1 is not important. At each day t, the goal is to find a distribution $\vec{p}^{(t)}$ over the experts so as to minimize the suffering (the cost). For each day t, the adversary decides that the cost of following the i'th expert's advice is $m_i^{(t)}$. Here, we have generalized the cost to be anything in $[-1, 1]$ instead of only counting the number of mistakes. (A negative number means that it was profitable to follow that expert's advice.) As we play a randomized strategy, the expected cost incurred at day t is thus

$$\sum_{i \in [N]} \Pr[\text{aggregator follows expert } i \text{ at day } t] \cdot m_i^{(t)} = \sum_{i \in [N]} p_i^{(t)} m_i^{(t)} =: \vec{p}^{(t)} \cdot \vec{m}^{(t)}\,.$$

The ultimate goal is to design a strategy that does as well as the best expert. That is, we wish to find a strategy such that for every $i \in [N]$

$$\underbrace{\sum_{t=1}^{T} \vec{p}^{(t)} \cdot \vec{m}^{(t)}}_{\text{our loss}} \leq \underbrace{\sum_{t=1}^{T} \vec{m}_i^{(t)}}_{\text{loss of best expert}} + (\text{small error terms})\,.$$

The Hedge strategy that we explain in the next section accomplishes this.

Example 1 *A natural example of the above situation is portfolio investment. Suppose you can allocate your money among 4 different funds: Postfinance, UBS, Credit Suisse, and Banque Cantonale. The distribution vector $\vec{p}^{(t)}$ at day t represents the fraction of your investment allocated to each fund. Your goal is to update these fractions over time so that your total return is nearly as good as the best single fund in hindsight. The observed cost (or profit) vector $\vec{m}^{(t)}$ reflects the daily performance of each fund. Remarkably, by continually rebalancing your portfolio according to the Hedge strategy, you can guarantee performance close to the best fund, up to a small error term. However, this guarantee becomes meaningful only when the time horizon T is sufficiently large; for small T, the error term may be significant.*

4 The Hedge strategy

We now explain and analyze the Hedge strategy for the setting introduced in the last section. It is parameterized by the "learning parameter" $\epsilon > 0$:

- Initially, assign each expert i a weight $w_i^{(1)}$ of 1. (All experts are equally trustworthy in the beginning.)

At each time t:

- Pick the distribution $p_i^{(t)} = w_i^{(t)}/\Phi^{(t)}$ where $\Phi^{(t)} = \sum_{i\in[N]} w_i^{(t)}$.
- After observing the cost vector, set $w_i^{(t+1)} = w_i^{(t)} \cdot e^{-\varepsilon \cdot m_i^{(t)}}$.

The difference between Hedge and the so-called Multiplicative Weight Update method is the weight-update $w_i^{(t+1)} = w_i^{(t)} \cdot e^{-\varepsilon \cdot m_i^{(t)}}$ instead of $w_i^{(t+1)} = w_i^{(t)} \cdot (1 - \varepsilon \cdot m_i^{(t)})$. Both methods have similar guarantees:

Theorem 2 *Suppose $\epsilon \leq 1$ and for $t \in [T]$, $\vec{p}^{(t)}$ is picked by Hedge. Then for any expert i,*

$$\sum_{t=1}^{T} \vec{p}^{(t)} \cdot \vec{m}^{(t)} \leq \sum_{t=1}^{T} m_i^{(t)} + \frac{\ln N}{\epsilon} + \epsilon T\,.$$

Note that in the above theorem, i can be chosen to be the best expert and thus Hedge does as well as the best expert plus some small error terms.

Proof Similar to the analysis of the weighted majority strategy, the proof goes by analyzing the potential $\Phi^t = \sum_{i\in[N]} w_i^{(t)}$.

Lower bound on $\Phi^{(T+1)}$: We lower bound the final potential as a function of i's performance:

$$\Phi^{(T+1)} = \sum_{j\in[N]} w_j^{(T+1)} \geq w_i^{(T+1)} = \exp(-\epsilon \sum_{t=1}^{T} m_i^{(t)})\,,$$

where the last equality follows from that the initial weight of i was one and every day t his weight was updated by $e^{-\epsilon m_i^{(t)}}$.

Upper bound on $\Phi^{(T+1)}$: We upper bound the final potential in terms of the total cost Hedge suffered. By definition,

$$\Phi^{(t+1)} = \sum_{j\in[N]} w_j^{(t+1)} = \sum_{j\in[N]} w_j^{(t)} \cdot \exp(-\epsilon m_j^{(t)})\,.$$

The exponentiated term is in $[-1, 1]$. Since $e^x \leq 1 + x + x^2$ for $x \in [-1, 1]$ (do a Taylor expansion),

$$\begin{aligned}
\sum_{j\in[N]} w_j^{(t)} \cdot \exp(-\epsilon m_j^{(t)}) &\leq \sum_{j\in[N]} w_j^{(t)} \left((1 - \epsilon m_j^{(t)} + \epsilon^2 \left(m_j^{(t)} \right)^2 \right) \\
&\leq \sum_{j\in[N]} w_j^{(t)} \left((1 - \epsilon m_j^{(t)} + \epsilon^2 \right) \qquad (\text{since } \left(m_j^{(t)} \right)^2 \leq 1) \\
&= \sum_{j\in[N]} w_j^{(t)} (1 + \epsilon^2) - \sum_{j\in[N]} \epsilon w_j^{(t)} m_j^{(t)} \\
&= \Phi^{(t)} (1 + \epsilon^2) - \epsilon \sum_{j\in[N]} \Phi^{(t)} p_j^{(t)} m_j^{(t)} \qquad (\text{since } \Phi^{(t)} p_j^{(t)} = w_j^{(t)}) \\
&= \Phi^{(t)} \left(1 + \epsilon^2 - \epsilon \vec{p}^{(t)} \vec{m}^{(t)} \right) \\
&\leq \Phi^{(t)} \cdot \exp \left(\epsilon^2 - \epsilon \vec{p}^{(t)} \vec{m}^{(t)} \right) \qquad (\text{since } 1 + x \leq e^x) \,.
\end{aligned}$$

We therefore have

$$\begin{aligned}
\Phi^{(T+1)} &\leq \Phi^{(T)} \cdot \exp \left(\epsilon^2 - \epsilon \vec{p}^{(T)} \vec{m}^{(T)} \right) \\
&\leq \Phi^{(T-1)} \cdot \exp \left(\epsilon^2 - \epsilon \vec{p}^{(T-1)} \vec{m}^{(T-1)} \right) \cdot \exp \left(\epsilon^2 - \epsilon \vec{p}^{(T)} \vec{m}^{(T)} \right) \\
&\quad \vdots \\
&\leq \Phi^{(1)} \cdot \exp \left(\epsilon^2 T - \epsilon \sum_{t=1}^{T} \vec{p}^{(t)} \vec{m}^{(t)} \right) \\
&= N \cdot \exp \left(\epsilon^2 T - \epsilon \sum_{t=1}^{T} \vec{p}^{(t)} \vec{m}^{(t)} \right) ,
\end{aligned}$$

where for the equality we used that $\Phi(1) = N$ since each expert was initialized with a weight of 1.

The above bounds give us

$$\exp(-\epsilon \sum_{t=1}^{T} m_i^{(t)}) \leq \Phi^{(T+1)} \leq N \cdot \exp \left(\epsilon^2 T - \epsilon \sum_{t=1}^{T} \vec{p}^{(t)} \vec{m}^{(t)} \right) .$$

Taking (natural) logarithms,

$$-\epsilon \sum_{t=1}^{T} m_i^{(t)} \leq \ln(N) + \epsilon^2 T - \epsilon \sum_{t=1}^{T} \vec{p}^{(t)} \vec{m}^{(t)}$$

and the final result follows by dividing by ϵ and rearranging the terms. ■

Remark The above proof may seem messy with all the inequalities. However, it follows a standard "framework": upper and lower bound the potential function. These Multiplicative Weight Update type of algorithms are most often analyzed using this potential argument. Once you know that cool argument, the analysis is almost automatic modulo some rewriting.

For some settings, it is convenient to consider the *average cost* incurred per day. We also generalize so that the cost vector can take values in $[-\rho, \rho]^N$ where ρ is called the "width". The proof is the same by scaling, and we get the following corollary:

Corollary 3 *Suppose $\epsilon \leq 1$ and for $t \in [T]$, $\vec{p}^{(t)}$ is picked by Hedge and cost vectors satisfy $\vec{m}^{(t)} \in [-\rho, \rho]^N$. If $T \geq (4\rho^2 \ln N)/\epsilon^2$, then for any expert i:*

$$\frac{1}{T}\sum_{t=1}^{T} \vec{p}^{(t)} \cdot \vec{m}^{(t)} \leq \frac{1}{T}\sum_{t=1}^{T} m_i^{(t)} + 2\epsilon\,.$$

References

[1] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta algorithm and applications. Technical report, Princeton University, 2005. 16

[2] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. In FOCS, pages 256–261, 1989. 16.1.1