CS168: The Modern Algorithmic Toolbox Lecture #17: Multiplicative Weights Update

Margalit Glasgow and Greg Valiant

1 Introduction

In this lecture, we'll consider a type of online decision making, where at each step $t = 1, \dots T$, you are presented with some data, and tasked with making a decision on the fly. Consider the following concrete setting, called the **Learning with Experts** problem.

In Learning with Experts, there are N experts, and each day $t = 1, \dots T$, each expert predicts the answer to a "Yes" or "No" question, such as "Will it rain today?". On each day, after seeing the expert opinions, the algorithm must make some prediction "Yes" or "No". The goal is to make as few mistakes as possible over the T days.

If all the experts are unhelpful, for example outputting random guesses at each step, then we cannot expect any algorithm that makes predictions based on the expert advice to output predictions better than with 50% accuracy. As we will see, however, it will be possible to perform comparably to best expert in hindsight, providing the time horizon, T is sufficiently large.

Our results will be formulated in terms of how well our algorithm performs, *in comparison to the performance of the best single expert.* Formally, we will consider the *regret,* defined as:

$$\text{Reg}(T) := \#\text{ mistakes made by alg} - \#\text{ mistakes made by best expert}. \qquad (1)$$

To gain some intuition for why achieving a small regret (say, one that scales sublinearly in T) might be possible, consider the setting where there is one expert that always predicts the correct answer, while all the other experts output a random answer. In this case, our algorithm could listen to the advice of any expert that has made no mistakes so far. After $\log(N)$ timesteps, we can expect that all the bad experts will have made a mistake, and thus for the remaining time, we will listen to the correct expert. In this case, we will achieve a regret of $\log(N)$, which doesn't even scale with T!

While we can't always achieve this good of a regret, a more general idea of up-weighting the opinion of experts that have made few mistakes, and down-weighting the opinion of experts that have made many mistakes turns out to work well. In this lecture, we'll see an algorithm called **Multiplicative Weights** which can achieve the following regret guarantee:

Theorem 1.1 *The Multiplicative Weights algorithm (MW) achieves:*

$$\#\textit{ mistakes made by MW} - \#\textit{ mistakes made by best expert} \leq 2\sqrt{T \log(N)}. \qquad (2)$$

One useful aspect of this result is that it works even if the correct answer at each step is adversarially chosen. That is, in the case of predicting rain, the skies would be able to decide to open or close depending on *both* the predictions of the experts and knowledge of the algorithm's strategy. We'll discuss this more later.

Applications The multiplicative weights update (MW) strategy which we will see in this lecture has been rediscovered many times in various fields of computer science and operations research.

Although we pitched MW as being motivated by online decision making, the applications of MW go far beyond what one might think of as online decision making. It turns out many problems can be solved via an online/iterative approach, where the solution is progressively improved by up-weighting various desiderata for a solution via MW, and using these weights to update the current solution. Its not always straightforward to see the connection between the Learning with Experts game and the application, since the objects that MW are applied to may not seem like "experts".

1. **Game Theory** One of the first uses of MW was in finding Nash equillibria of two-player zero-sum games. Here the game is represented by a $m \times n$ payoff matrix, and the row (resp. column) player must choose a distribution over the m (resp n) strategies. Here MW is used in a alternating fashion by the two players to update their distributions over strategies, which in this case are the "experts".

2. **Learning Theory.** The most well-known use for MW is learning theory is the *Adaboost* algorithm, which is a way designing a strong binary classifier (say that classifies the data with 90% accuracy) from many weak classifiers (which perform only slightly better than a random guess). Prior to the discovery of Adaboost, MW was used to design the Winnow algorithm for linear classification. In both algorithms, the classifier is learned in an iterative fashion, where at each step, the data samples that are misclassified are up-weighted using MW, and the classifier is updated to better fit the new distribution over data samples.

3. **Optimization.** The MW algorithm can be used to solve various convex optimization problems, including linear programs. We will see this in Lecture #18. Here the MW algorithm is used to track a distribution over constraints, where the violated constraints are up-weighted.

4. **Bandits** Variants of MW (see the Exp3 algorithm) can be used in bandit feedback setting where you don't get to see the loss of each expert ("arm") at every round, but rather only the loss of the expert which the algorithm chose to follow.

2 Learning with Experts: Formally & More General Setting

Recall the Learning with Expert problem we introduced.

Learning with Experts Game.
For $t = 1, \dots, T$:

1. The experts are presented with a "Yes" or "No" question.
2. Each expert predicts an answer "Yes" or "No".
3. The algorithm chooses a distribution p_t (so $p_t(i)$ = probability of listening to expert i).
4. The adversary reveals some correct answer "Yes" or "No" to the question.
5. The algorithm samples an expert according to the distribution p_t, and follows that expert's prediction.

The total regret is the total number of mistakes made by the algorithm minus the number of mistakes made by the best expert.

More generally, learning from experts can capture a setting beyond binary "Yes" or "No" advice, where instead of each expert either being correct or incorrect at step t, each expert i incurs some loss $\ell_t(i)$ which is bounded in $[-1, 1]$. The Yes/No experts is a special case of this where the loss is 1 if the expert made a mistake, and 0 otherwise.

General Learning with Experts Game.
For $t = 1, \dots, T$:

1. The algorithm chooses a distribution p_t (so $p_t(i)$ = probability of listening to expert i).
2. The adversary reveals some loss on all the experts $\ell_t(i)$ for $i \in 1, \dots, N$, which may depend on p_t.
3. The algorithm incurs the loss $\mathbb{E}_{i \sim p_t} \ell_t(i)$

The total regret is $\sum_{t=1}^{T} \mathbb{E}_{i \sim p_t} \ell_t(i) - \min_{i \in [N]} \sum_{t=1}^{T} \ell_t(i)$.

3 Algorithms for Learning with Experts

We'll work through a few candidate ideas for Learning with Experts to gain some intuition for the MW algorithm.

Follow-the-leader (FTL) algorithm. Choose the expert who has made the least mistakes (or incurred the least loss) up until time t.

Problem: Loss function could be chosen such that the algorithm *always* messes up. Eg., consider the following round-robin loss function. At time $t = 0$, the loss is i/N on expert i. For $t > 1$, the loss is given by

$$\ell_t(i) = \begin{cases} 1 & t = i + 1 \mod N \\ 0 & \text{otherwise.} \end{cases}$$

Then after step t, expert $t \mod N$ will have the least cumulative loss, so FTL will choose expert $t - 1 \mod N$ at step t, incurring a loss of 1. Thus the total loss incurred by the algorithm after T steps is T, while the best possible loss in hindsight is at most $\frac{T}{N} + 1$. This does not lead to sublinear regret.

Weight Experts Linearly by their Performance. Choose expert i with probability

$$p_t(i) \propto \#\text{correct predictions by expert } i \text{ up to time } t - 1\ . \qquad (3)$$

Problem: the weight on bad experts doesn't decay quickly enough. Eg., suppose we were in the Yes/No prediction setting with 2 experts, and one expert had success rate 100%, while the second had success rate 50%. Then on average, we will pick the bad expert with probability $\frac{1/2}{1+1/2} = \frac{1}{3}$, leading to roughly $T/6$ mistakes, which is large relative to the best expert which makes 0 mistakes.

Multiplicative Weights Update. The above strategies suggests that we should: (1) Use a randomized strategy and (2) more aggressively dismiss experts with large losses.

The multiplicative weight algorithm will maintain a vector $w_t \in \mathbb{R}^N$ which tracks the relative weights of the algorithm's confidence in each expert. At each step t, we will decay the weight *multiplicatively* with the loss attained by each expert. The strength of the decay is governed by a parameter ϵ.

Multiplicative Weights Update (MW).
Initialize $w_t(i) = 1 \quad \forall\, i \in [N]$.
For $t = 1, \ldots, T$:

1. Let $p_t(i) = \frac{w_t(i)}{\Phi_t}$, where $\Phi_t := \sum_i w_t(i)$.
2. Observe losses $\ell_t(i)$, and update $w_{t+1}(i) = w_t(i)e^{-\epsilon \ell_t(i)}$ for each $i \in [N]$.

Before analyzing the above algorithm, we first consider a related, simpler algorithm/analysis which applies to the setting of binary (Yes/No) predictions. In this simpler setting, we will analyze the **Weighted Majority** algorithm applied to weights $w_t(i) = 2^{-\text{\#experts i's mistakes}}$— namely where the weight of expert i is halved each time expert i makes a mistake. While this doesn't achieve as good of a bound as the more general MW algorithm, we will analyze it first since the proof has the same structure though is simpler/cleaner.

4 Analysis of Weighted Majority

The following theorem guarantees that the number of mistakes of the weighted algorithm is at most a small factor times the number of mistakes of the best expert.

Theorem 4.1 *The weighted majority algorithm achieves the following guarantee, for* every *expert i,*

$$\#\textit{mistakes weighted majority} \leq 2.41(\#\textit{mistakes expert } i + \log_2(N)) \tag{4}$$

Proof: To prove this theorem we will use a potential function argument, which tracks the value of the potential function $\Phi_t = \sum_{i=1}^N w_t(i)$.

The idea of our argument will be to show that if weighted majority (WM) makes a mistake, then Φ_t will have to decay significantly. However, since Φ_t must be at least as large as $w_t(i)$ for each i, if there is an expert i with good performance, $w_T(i)$, and thus Φ_T will be relatively large. We will use these two points together to show that MW cannot make too many mistakes.

More formally, observe the following:

1. $\Phi_1 = N$, since all weights start at 1.
2. $w_T(i) = \left(\frac{1}{2}\right)^{\text{\#i's mistakes}}$.
3. If WM makes a mistake at step t, then $\Phi_{t+1} \leq \frac{3}{4}\Phi_t$. This is because at least half of the total weight of Φ_t corresponds to experts that were wrong. So at least half of the weight gets halved.

Telescoping the 3rd bullet, and then plugging in (1) we have

$$\Phi_T \leq \Phi_1 \left(\frac{3}{4}\right)^{\text{\#WM's mistakes}} = N \left(\frac{3}{4}\right)^{\text{\#WM's mistakes}}. \tag{5}$$

Thus since $\Phi_T \geq w_T(i)$ for any i, we have

$$\left(\frac{1}{2}\right)^{\text{\#i's mistakes}} \leq N \left(\frac{3}{4}\right)^{\text{\#WM's mistakes}}. \tag{6}$$

Taking log (base 2) on both sides, we have

$$-\#\text{i's mistakes} \leq \log_2(N) + \log_2\left(\frac{3}{4}\right)\#\text{WM's mistakes} \tag{7}$$

Rearranging yields the theorem. ■

5 Analysis of Multiplicative Weights Update

The following theorem guarantees that the total regret of the MW algorithm relative to the loss of the best expert scales with $2\sqrt{T\log(N)}$. The proof follows the high level proof of the simplified version above.

Theorem 5.1 *The MW algorithm with parameter $\epsilon \leq 1$ achieves*

$$\sum_{t=1}^{T} \mathbb{E}_{i\sim p_t}\ell_t(i) - \min_{i\in[N]}\sum_{t=1}^{T}\ell_t(i) \leq T\epsilon + \frac{\log(N)}{\epsilon}. \tag{8}$$

Setting $\epsilon = \frac{\sqrt{\log(N)}}{\sqrt{T}}$ yields a regret of $2\sqrt{T\log(N)}$.

Proof: To prove this theorem we will use the same potential function argument as in the binary case, which tracks the value of $\Phi_t = \sum_{i=1}^{N} w_t(i)$.

The idea of the argument is the same as in the binary case above: if MW obtains a high loss, then Φ_t will have to decay significantly. However, if there is an expert i with good performance, than Φ_t must be at least as large as $w_t(i)$, that is

$$w_t(i) = \prod_{t=1}^{T}\exp(-\epsilon\ell_t(i)) = \exp\left(-\epsilon\sum_{t=1}^{T}\ell_i(t)\right). \tag{9}$$

Thus MW cannot obtain a high loss too many times.

To formalize this, we consider how much Φ_t decays each step. For each t, we have

$$\Phi_{t+1} = \sum_{i=1}^{N} w_{t+1}(i) \tag{10}$$

$$= \sum_{i=1}^{N} w_t(i)\exp(-\epsilon\ell_t(i)) \tag{11}$$

$$\tag{12}$$

Since $e^{-x} \leq 1 - x + x^2$, this is at most

$$\sum_{i=1}^{N} w_t(i) \left(1 - \epsilon \ell_t(i) + \epsilon^2 \ell_t(i)^2\right) \tag{13}$$
$$\leq \sum_{i=1}^{N} w_t(i) \left(1 - \epsilon \ell_t(i) + \epsilon^2\right) \tag{14}$$
$$= (1+\epsilon^2) \sum_{i=1}^{N} w_t(i) - \epsilon \sum_{i=1}^{N} w_t(i) \ell_t(i) \tag{15}$$
$$= (1+\epsilon^2)\Phi_t - \epsilon \Phi_t \sum_{i=1}^{N} p_t(i) \ell_t(i) \tag{16}$$
$$= \Phi_t \left(1 + \epsilon^2 - \epsilon \langle p_t, \ell_t \rangle\right) \tag{17}$$
$$\leq \Phi_t \exp(\epsilon^2 - \epsilon \langle p_t, \ell_t \rangle) \tag{18}$$

Here we have denoted $\langle p_t, \ell_t \rangle = \sum_{i=1}^{N} p_t(i)\ell_t(i)$, and in the last line, we used the fact that $1 - x \leq e^{-x}$.

Summarizing, we have the following bound on the decay of Φ_t at each step:

$$\Phi_{t+1} \leq \Phi_t \exp(\epsilon^2 - \epsilon \langle p_t, \ell_t \rangle). \tag{19}$$

Telescoping bound over all T steps, we have that

$$\Phi_T \leq \Phi_1 \prod_{t=1}^{T} \exp(\epsilon^2 T - \epsilon \langle p_t, \ell_t \rangle) = \Phi_1 \exp\left(\epsilon^2 T - \epsilon \sum_{t=1}^{T} \langle p_t, \ell_t \rangle\right) \tag{20}$$
$$= N \exp\left(\epsilon^2 T - \epsilon \sum_{t=1}^{T} \langle p_t, \ell_t \rangle\right). \tag{21}$$

Here in the second line we plugged in $\Phi_1 = N$, since all the weights start at 1.

Now recall from Equation 9 that for any expert i, we have $\Phi_t \geq w_t(i)$. Combining this with the equation above, we have

$$\exp\left(-\epsilon \sum_{t=1}^{T} \ell_i(t)\right) \leq \Phi_T \leq N \prod_{t=1}^{T} \exp(\epsilon^2 T - \epsilon \langle p_t, \ell_t \rangle) \tag{22}$$

Taking logarithms yields

$$-\sum_{t=1}^{T} \ell_i(t) \leq \log(N) + \epsilon^2 T - \epsilon \sum_{t=1}^{T} \langle p_t, \ell_t \rangle, \tag{23}$$

and thus

$$\sum_{t=1}^{T} \langle p_t, \ell_t \rangle - \sum_{t=1}^{T} \ell_i(t) \leq \epsilon T + \frac{\log(N)}{\epsilon}, \tag{24}$$

as desired. ■

Remark 5.2 If the losses are instead bounded between $[-\rho, \rho]$, then by scaling down the losses by a factor of ρ, we achieve the guarantee

$$\sum_{t=1}^{T} \langle p_t, \ell_t \rangle - \sum_{t=1}^{T} \ell_i(t) \leq \rho \epsilon T + \frac{\rho \log(N)}{\epsilon}. \tag{25}$$