Advanced Algorithms 12/19, 2017

Lecture 14

Lecturer: Mohsen Ghaffari *Scribe: Ahad N. Zehmakan*

This draft has not been checked by the lecturer yet.

1 The multiplicative weights update method

The *multiplicative weights update method*, also known as *learning from the experts* is a very useful technique applicable to a wide variety of learning and optimization problems; for instance, see [PST95] regarding approximation solutions for fractional packing and covering problems. Algorithms in this setting maintain a probability distribution over a certain set that is updated iteratively by a multiplicative rule. The deterministic and randomized variants of the method are discussed respectively in Section 1.1 and Section 1.2, and finally in Section 1.3 a more general framework is argued.

1.1 Deterministic Weighted Majority

Consider the following simple example. There are n experts $e_1, \cdots, e_n$ who predict the stock market every day. The predictions of the experts are binary valued (zero/one). Assume one of the experts is perfect, meaning s/he never makes a mistake. An online learning algorithm sees the predictions of the experts every day and makes a prediction of its own and its goal is to minimize the total number of mistakes made. Now, consider the following simple majority-based algorithm: by starting from $E = \{e_1, \cdots, e_n\}$, in each round (day) it predicts by following the most frequent prediction among experts in E and removes the experts who predicted incorrectly from E. This algorithm does not make more than $\log n$ mistakes, no matter how many iterations it is applied, because every time it makes a mistake at least half of the experts get removed from E. Notice, the existence of a perfect expert guarantees the non-emptiness of set E.

The aforementioned algorithm enjoys the unrealistic assumption that there exists a perfect expert; in other words, it eventually fails (the set E will be empty) if there is no perfect expert. However, the main idea behind the algorithm is to update some initially associated weights to the experts depending on the correctness of their prediction, where the update rule simply is to multiply the weight of all wrong experts by zero. One might modify the algorithm to cover also the setting in which there is no perfect expert. Experts making wrong predictions should not be dropped, but their weights can be reduced by a constant factor, say $1-\epsilon$ for $0 < \epsilon < 1/2$. More precisely, by starting from $w_i = 1$ for $1 \leq i \leq n$, where w_i denotes the weight corresponding to expert e_i, in each round the algorithm predicts according to the weighted majority of the experts. Furthermore, it sets $w_i = (1-\epsilon)w_i$ if expert e_i's prediction is wrong.

If the algorithm makes a mistake, the weight of the wrong experts is at least half the total weight. The weight of the wrong experts gets reduced by $1-\epsilon$, so the total weight is reduced by a factor of at least $1-\epsilon/2$ for every mistake. After making m mistakes, the total weight is at most $n(1-\epsilon/2)^m$. On the other hand, obviously the total weight is lower-bounded by $(1-\epsilon)^{m^*}$, where m^* denotes the number of mistakes made by the best expert. Putting the lower bound and the upper bound together and taking logarithms we have

$$(1-\epsilon)^{m^*} \leq n(1-\epsilon/2)^m \Rightarrow m^* \ln(1-\epsilon) \leq \ln n + m \ln(1-\epsilon/2).$$

By applying $-\epsilon - \epsilon^2 \leq \ln(1-\epsilon) < -\epsilon$ (see Lemma 1), we have $-m^*(\epsilon+\epsilon^2) \leq \ln n - m\epsilon/2$. Finally, by rearranging and dividing by $\epsilon/2$, we have $m \leq 2(1+\epsilon)m^* + \frac{2\ln n}{\epsilon}$.

Therefore, the weighted majority algorithm can perform acceptably in the sense that the number of its mistakes is not more than roughly two times the number of mistakes by the best expert plus some logarithmic term in n. Actually, the following simple example demonstrates that it is almost the best that one can hope for; i.e., it is not possible to achieve a constant better than 2. Suppose there are two experts e_1, who always predicts 1, and expert e_2, who always predicts 0. In the wost-case scenario, no matter how we break a tie, all our prediction can go wrong while at least one of the two experts is right in at least fifty percent of the situations.

Lemma 1. *For* $0 < \epsilon < 1/2$, $-\epsilon - \epsilon^2 \leq \ln(1-\epsilon) < -\epsilon$.

Proof. The Taylor series expansion for $\ln(1-\epsilon)$ is given by $\ln(1-\epsilon) = \sum_{i\in\mathbb{N}} -\epsilon^i/i$. From the expansion we have $\ln(1-\epsilon) < \epsilon$ as the discarded terms are negative.

The other half is equivalent to the inequality $1-\epsilon \geq e^{-\epsilon-\epsilon^2}$. By the convexity of the function $e^{-\epsilon-\epsilon^2}$, the inequality is true for all ϵ less than a threshold ϵ_t. Substituting $\epsilon = 1/2$ we have $e^{-3/4} < 1/2$ showing that the threshold ϵ_t is more than $1/2$. □

1.2 Randomized Weighted Majority

So far, we argued the deterministic approach and observed we cannot have a guarantee factor better than two, in comparison to the best expert. Does the randomness allow us to achieve a better performance? The answer is yes. More accurately, we show a probabilistic strategy, which chooses experts with probabilities proportional to their weights, makes at most $(1+\epsilon)m^* + \ln n/\epsilon$ mistakes in expectation.

At the beginning, all experts have weight 1; i.e., $w_i = 1$ for $1 \leq i \leq n$. In each round, the algorithm chooses one of the experts at random with probability proportional to his/her weight and follows his/her advice. Furthermore, it changes the weight w_i to $(1-\epsilon)w_i$ for each expert e_i who predicts wrongly in this round. Let random variable m denote the number of mistakes made by the algorithm.

Theorem 2. *For* $0 < \epsilon < 1/2$, $\mathbb{E}[m] \leq (1+\epsilon)m^* + \ln n/\epsilon$.

Proof. Let F_i be the weighted fraction of experts that are wrong in round i; clearly, $\mathbb{E}[m] = \sum_i F_i$. Let Φ_i denote the sum of the weights in the i-th round. Then, $\Phi_i \leq \Phi_i(1-\epsilon F_i)$. The sum of the weights in round T is upper-bounded by $n\prod_{i=1}^{T}(1-\epsilon F_i)$. Assuming the best expert makes m^* mistakes in the first T rounds and utilizing the estimate $1-x \leq e^{-x}$, we have

$$(1-\epsilon)^{m^*} \leq n\prod_{i=1}^{T}(1-\epsilon F_i) \Rightarrow (1-\epsilon)^{m^*} \leq ne^{-\sum_{i=1}^{T}\epsilon F_i}.$$

Taking logarithms and applying Lemma 1 result in

$$m^*\ln(1-\epsilon) \leq \ln n - \epsilon\mathbb{E}[m] \Rightarrow \mathbb{E}[m] \leq -\frac{\ln(1-\epsilon)}{\epsilon}m^* + \frac{\ln n}{\epsilon} \leq (1+\epsilon)m^* + \frac{\ln n}{\epsilon}.$$

□

Note that as ϵ becomes smaller, our multiplicative overhead over m^* is approaching 1, but the additive term is growing. So, we usually want to choose ϵ in a way that balances out these two effects.

1.3 General Framework

Now, we discuss that the aforementioned method can be extended to capture a more general framework. Assume the experts can choose from an arbitrary space of decisions and at the end of each round t a "loss" $l_i^t \in [-\rho, \rho]$ for expert e_i, $1 \leq i \leq n$, is revealed. Note that we allow l_i^t to be negative; i.e., they can correspond to gains. A natural generalization of our algorithm from Section 1.2 would be to initially assign weight $w_i = 1$ to the expert e_i for $1 \leq i \leq n$ and in each round the algorithm picks one of the experts at random with probability proportional to his/her weight. Furthermore, the algorithm updates $w_i = (1 - \frac{\epsilon}{\rho}l_i^t)w_i$.

Observe that in the case when $\rho = 1$ and l_i^t is either 0 (for correct prediction) or 1 (for wrong prediction) we are back in the same setting as Section 1.2. Very similar to the proof of Theorem 2, one can show[1] in the general setting the expected loss of this algorithm is upper bounded by $\sum_t l_i^t + \sum_t |l_i^t| + \frac{\rho\ln n}{\epsilon}$. Note that as this result allows us to compare our performance with respect to an arbitrary expert then in particular it has to hold for the best one among them.

References

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

[PST95] Serge A Plotkin, David B Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20(2):257–301, 1995.

[1] For a detailed proof, see Theorem 2.1 in [AHK12].