COMPSCI 638: Graph Algorithms	November 1, 2019

Lecture 20

Lecturer: Debmalya Panigrahi — *Scribe: Kevin Sun*

1 Overview

In this lecture, we give an introduction to the multiplicative weight update method. This is a general algorithmic technique that has applications in various fields and problems.

2 Weighted Majority Algorithms

Consider the following problem: at each time step $t = 1, \ldots, T$, we must predict a binary outcome, and we have access to n experts, each of whom offers an opinion. Once we've made a prediction, we incur some loss ℓ^t: if our decision was correct, then $\ell^t = 0$, and otherwise, $\ell^t = 1$. Our performance is measured against the number of errors the best expert (in hindsight) made over all T steps. (Note that it is possible that we incur fewer losses than the best expert.)

We first establish some notation. Let $L^t = \sum_{t'=1}^{t} \ell^{t'}$ denote the total loss that we have incurred through time t, $L = L^T$ denote the total loss, i^* denote the expert that incurred the least total amount of loss, and L^* denote the total loss of i^*. Thus, our goal is to bound L against L^*.

2.1 Deterministic Weighted Majority

We begin by presenting the basic, deterministic version of the weighted majority algorithm. The idea is the following: initially, we simply poll the experts and predict the majority outcome. After the outcome is revealed, we decrease the weight of experts that were incorrect, so that in subsequent steps, their opinion is slightly discounted.

Algorithm 1 Weighted Majority Algorithm

```
Initialize w_i^0 = 1 for i ∈ {1, ..., n} and fix some ε ∈ (0, 1/2).
for t = 1, ..., T do
    Output the weighted majority opinion, where expert i is given weight w_i^{t-1}.
    for i = 1, ..., n do
        if ℓ_i^t = 1 (i.e., expert i was incorrect then
            w_i^t = (1 − ε) w_i^{t-1}
        else
            w_i^t = w_i^{t-1}
```

Theorem 1. *The loss L incurred by Algorithm 1 satisfies the following:*

$$L \leq 2(1+\epsilon)L^* + \frac{2 \ln n}{\epsilon}.$$

Proof. Let $W^t = \sum_i w_i^t$ denote sum of the weights on the experts at time t, so initially, we have $W^0 = n$. Whenever $\ell^t = 1$, then at least half of the total weight gets reduced by a $(1-\epsilon)$ factor, so

$$W^t \leq \left(1 - \frac{\epsilon}{2}\right) W^{t-1}.$$

Thus, since our total loss is L, the total weight at the end is

$$W^T \leq \left(1 - \frac{\epsilon}{2}\right)^L n. \tag{1}$$

Now consider the weight of the best expert i^*, who incurs a total loss of L^*:

$$w_{i^*}^T = (1-\epsilon)^{L^*}. \tag{2}$$

Combining (1) and (2) with $W^T \geq w_{i^*}^T$, and taking a natural logarithm, yields

$$L \ln\left(1 - \frac{\epsilon}{2}\right) + \ln n \geq L^* \ln(1-\epsilon).$$

We now apply the inequality $-x - x^2 \leq \ln(1-x) \leq -x$ when $x \in (0, 1/2)$ to get

$$L\left(-\frac{\epsilon}{2}\right) + \ln n \geq L^*(-\epsilon - \epsilon^2),$$

and rearranging this yields the desired inequality. □

Remark: We note that the factor of 2 obtained in Theorem 1 is unavoidable: consider the setting where there are two experts that disagree at every step, and at every step, our algorithm incurs a loss for a total loss of T. The experts, together, incur a total loss of T, so the best expert incurs at most $T/2$ loss. But this construction assumes the algorithm is deterministic, i.e., the input can be chosen adversarially. So now, we show how to overcome this 2 using a randomized algorithm.

2.2 Randomized Weighted Majority

The algorithm is a simple modification of Algorithm 1: at each step, we flip a coin with bias proportional to the total weight of each outcome.

Theorem 2. *The randomized weighted majority algorithm has expected loss*

$$L \leq (1+\epsilon)L^* + \frac{\ln n}{\epsilon}.$$

Proof. The proof is similar to the proof of Theorem 1, and we follow the same notation. But now, we let ℓ^t denote the *expected* loss at step t, so ℓ^t is the proportion of weight assigned to the incorrect outcome. Thus,

$$W^t \leq (1 - \epsilon\ell^t) W^{t-1}.$$

Thus, the total weight after T time steps is

$$W^T \leq \left(\prod_{i=1}^{T} (1 - \epsilon\ell^t)\right) n$$

Applying $W^T \geq w_{i^*}^T = (1-\epsilon)^{L^*}$ to the previous inequalities yields

$$\left(\prod_{i=1}^{T}(1-\epsilon\ell^t)\right) n \geq (1-\epsilon)^{L^*}$$

Taking the natural logarithm of both sides yields

$$\sum_{t=1}^{T}\ln\bigl(1-\epsilon\ell^t\bigr) + \ln n \geq L^* \ln(1-\epsilon), \tag{3}$$

Again, applying the bound $\ln(1-x) \leq -x$ gives us

$$\sum_{t=1}^{T}\ln\bigl(1-\epsilon\ell^t\bigr) \leq -\epsilon\sum_{t=1}^{T}\ell^t = -\epsilon L.$$

Substituting this into (3) and rearranging yields the desired inequality. □

2.3 Generalized Loss

We now generalize our problem setting: our loss at step t can now be any real number $\ell^t \in [0,1]$, and at each step t, we let $\ell_i^t \in [0,1]$ denote the loss of expert i. Our algorithm is still the randomized weighted majority algorithm from Sec. 2.2, but we generalize our update rule to the following, which occurs at every step t for every expert i.

$$w_i^t = (1-\epsilon\ell_i^t)w_i^{t-1}.$$

(Updating the weights by a factor of $(1-\epsilon)^{\ell_i^t}$ is essentially equivalent.) At each step, we make a decision by following an expert chosen with probability proportional to their weight.

Theorem 3. *For generalized loss $\ell^t \in [0,1]$, the above algorithm incurs loss*

$$L \leq (1+\epsilon)L^* + \frac{\ln n}{\epsilon}.$$

Proof. We follow the notation as the proof of Theorem 2 to obtain

$$W^T \leq \left(\prod_{t=1}^{T}(1-\epsilon\ell^t)\right) n.$$

Furthermore, the final weight of the best expert i^* is the following:

$$w_{i^*}^T = \prod_{t=1}^{T}(1-\epsilon)^{\ell_{i^*}^t} = (1-\epsilon)^{L^*}.$$

The rest of the proof is identical to the proof of Theorem 2. □

Remark: We can further generalize the problem by assuming the loss $\ell^t \in [0,p]$ for some $p \geq 1$. Then scaling the entire problem down by p allows us to apply Theorem 3 to the loss quantities L/p and L^*/p. Thus, scaling back to the original problem gives us a loss guarantee of

$$L \leq (1+\epsilon)L^* + \frac{p\ln n}{\epsilon}.$$

This quantity p is the known as *width* of the problem, and in the next lecture, we will see its importance in the design of algorithms.

3 Summary

In this lecture, we began our study of the multiplicative weight update method by looking at the weighted majority algorithm and its randomized variant. We also studied a generalized loss function and introduced the notion of width, which will be used in future lectures.