CS 6840 Algorithmic Game Theory September 6, 2024

Lecture 5: Randomized Multiplicative Weights (Hedge)

Instructor: Eva Tardos *Scribe: David Lin*

This lecture was given by Bobby Kleinberg.

Last time, we saw the WEIGHTED MAJORITY algorithm for the binary labeling with expert advice problem. The WEIGHTED MAJORITY is a determinstic algorithm, parameterized by $\epsilon \in (0, \frac{1}{2})$, satisfying a mistake bound of

$$m_{\text{ALG}} \leq \frac{2\ln(K)}{\epsilon} + 2(1+\epsilon)m_{\text{OPT}},$$

where K is the number of experts and m_{OPT} is the number of mistakes by the most accurate single expert.

The factor of 2 in the term $2(1+\epsilon)m_{\text{OPT}}$ is tight for determinstic algorithms, in that any determinstic algorithm will make at least $2m_{\text{OPT}}$ mistakes in the worst case. To see this, consider the scenario where there are two experts, one always predicting 0 and one always predicting 1. For any determinstic algorithm, we can simulate the algorithm while creating a binary sequence such that the determinstic algorithm always predicts the binary label incorrectly. But at least one of the experts will always get a majority of the predictions correct, so $m_{\text{ALG}} \geq 2m_{\text{OPT}}$ in this case.

Today, we will see a randomized algorithm called HEDGE that shaves off that factor of 2. Sometimes this algorithm is called the multiplicative weights algorithm, but multiplicative weights is sometimes used as a more general term to refer to a broader family of algorithms.

The setting for the HEDGE algorithm is as follows.

- There are K experts.
- Time progresses in timesteps $t = 1, 2, \ldots, T$. At each time t, the algorithm selects a probability distribution p_t over the experts $[K]$ and an adversary selects a loss vector $l_t \in [0,1]^K$. The probability distribution p_t and loss vector l_t can depend on the past history $(p_s, l_s)_{s=1}^{t-1}$ but p_t and l_t are chosen simulatenously.
- The loss of expert i is $l_t(i)$. The algorithm's loss is $\langle l_t, p_t\rangle = \sum_{i=1}^{k} l_t(i)p_t(i)$.

How does this relate to the binary prediction problem of last lecture? The binary prediction problem can be thought of in the context of this problem by defining the loss vector as

$$l_t(i) = \begin{cases} 1 & \text{if expert } i \text{ made a mistake at time } t \\ 0 & \text{otherwise} \end{cases}$$

and forcing p_t to be a determinstic choice of experts $[K]$ as opposed to a probability distribution over $[K]$. The problem today generalizes the binary prediction problem by allowing l_t to be an arbitrary real-valued vector in $[0,1]^K$ as opposed to a binary one, and allows the algorithm to choose a probability distribution p_t over the experts as opposed to just predicting a binary label. Note that this reduction does not fully encapsulate the binary prediction problem from last time. In the binary prediction problem, the algorithm guessed either 1 or 0, potentially using expert advice, while in this problem, p_t does not guess 1 or 0 explicitly but rather it chooses a probability distribution over the experts.

Now we give HEDGE algorithm. It is parameterized by $\epsilon \in (0, \frac{1}{2})$. Define $L_t = \sum_{s=1}^{t} l_s$ to be total loss up until time t. At each time t, the HEDGE algorithm sets weights $w_t(i) = (1-\epsilon)^{L_{t-1}(i)}$. Define the total weight $W_t = \sum_{i=1}^{k} w_t(i)$. The algorithm predicts the probability distribution p_t given by $p_t(i) = \frac{w_t(i)}{W_t}$

This is similar to the WEIGHTED MAJORITY algorithm: if the losses $l_t(i)$ correspond to whether or not expert i made a mistake, then $L_{t-1}(i)$ is the total number of incorrect answers, and the weight update rule is exactly the same as WEIGHTED MAJORITY.

Let's analyze the HEDGE algorithm. As in the analysis of the WEIGHTED MAJORITY algorithm, we will compare the algorithm's loss with the decrease in the total weight W_t, which we will compare to the best expert's loss.

We have

$$\begin{aligned} w_{t+1}(i) &= (1-\epsilon)^{L_t(i)} = (1-\epsilon)^{L_{t-1}(i)+l_t(i)} \\ &= (1-\epsilon)^{l_t(i)}(1-\epsilon)^{L_{t-1}(i)} = (1-\epsilon)^{l_t(i)} w_t(i) \\ &\le (1-\epsilon l_t(i)) w_t(i), \end{aligned}$$

using the fact that $(1-\epsilon)^x \le 1-\epsilon x$ for all $x \in [0,1]$ for the inequality. This implies

$$\begin{aligned} W_{t+1} &= \sum_{i=1}^{K} w_{t+1}(i) \\ &\le \sum_{i=1}^{k} (1-\epsilon l_t(i)) w_t(i) \\ &= \sum_{i=1}^{K} w_t(i) - \epsilon \sum_{i=1}^{K} l_t(i) w_t(i) \\ &= W_t - \epsilon \sum_{i=1}^{k} l_t(i) p_t(i) W_t \\ &= W_t(1-\epsilon\langle l_t, p_t\rangle). \end{aligned}$$

Taking logarithms, we obtain

$$\begin{aligned} \ln W_{t+1} &\le \ln W_t + \ln(1-\epsilon\langle l_t, p_t\rangle) \\ &\le \ln W_t - \epsilon\langle l_t, p_t\rangle, \end{aligned}$$

using the identity $\ln(1-x) \le -x$. Combining inequalities inductively, we obtain

$$\ln W_{T+1} \le \ln W_0 - \epsilon \sum_{t=1}^{T} \langle l_t, p_t\rangle = \ln K - \epsilon \sum_{t=1}^{T} \langle l_t, p_t\rangle.$$

Let i^* be the best expert so that $L_T(i^*) = \min_{i\in[K]} L_T(i)$. Then

$$W_{T+1} = \sum_{i=1}^{K} (1-\epsilon)^{L_T(i)} > (1-\epsilon)^{L_t(i^*)}.$$

Taking logarithms,

$$\begin{aligned} & \ln W_{T+1} > \ln(1-\epsilon) L_T(i^*) \\ \Longrightarrow\ & \ln(1-\epsilon) L_T(i^*) < \ln K - \epsilon \sum_{t=1}^{T} \langle l_t, p_t\rangle \\ \Longrightarrow\ & \epsilon \sum_{t=1}^{T} \langle l_t, p_t\rangle < \frac{\ln K}{\epsilon} - \frac{\ln(1-\epsilon)}{\epsilon} L_T(i^*) \end{aligned}$$

$$\implies \sum_{t=1}^{T} \langle l_t, p_t \rangle < \frac{\ln K}{\epsilon} + \frac{\epsilon + \epsilon^2}{\epsilon} L_T(i^*) = \frac{\ln K}{\epsilon} + (1+\epsilon) L_T(i^*),$$

using the identity $-\ln(1-\epsilon) \leq \epsilon + \epsilon^2$ for $\epsilon \in (0, \frac{1}{2})$. This bound's the algorithm's loss in terms of the best expert's loss.

How do we choose ϵ? Like with WEIGHTED MAJORITY, it is a choice for the algorithm designer. One way to do so is by attempting to minimize regret, a very important notion in learning theory, where regret is defined as the difference in the expected loss of the algorithm and the best expert's loss:

$$R^{\text{ALG}}(T) = \sum_{t=1}^{T} \langle l_t, p_t^{ALG} \rangle - L_T(i^*).$$

For HEDGE, we have the regret bound

$$R^{\mathsf{HEDGE}}(T) < \frac{\ln K}{\epsilon} + \epsilon L_T(i^*).$$

We can try to minimize this, but we don't know $L_T(i^*)$. We will see two ways of getting around this issue.

One way is trying to use the obvious bound $L_T(i^*) \leq T$, so we obtain

$$R^{\mathsf{HEDGE}}(T) < \frac{\ln K}{\epsilon} + \epsilon T.$$

By elementary calculus, this is minimized when $\epsilon = \sqrt{\frac{2 \ln K}{T}}$, in which case we have the bound

$$R^{\mathsf{HEDGE}}(T) < \sqrt{2 \ln(K) T}.$$

The $\sqrt{T}$ factor is very typical of well-designed learning algorithms. In fact, in this problem, any algorithm can have at least $\Omega(\sqrt{T})$ regret. Consider again the binary prediction scenario with two experts, one always predicting 0 and another predicting 1. For each t, let the correct label be the result of an independent coin flip. Any algorithm must suffer $\frac{T}{2}$ loss in expectation. But the regret of the best expert has expected loss $\frac{T}{2} - \Theta(\sqrt{T})$ (given by properties of the binomial distribution). Hence, any algorithm has $\Theta(\sqrt{T})$ regret here.

Another way to choose ϵ is based on what's known as the "doubling trick". At time t, we don't what $L_T(i^*)$ is, but we do know the loss of the current best expert, $\min_{i \in [K]} L_t(i)$. We modify the algorithm as follows.

- We start by assuming $L_T(i^*)$ will be at most 1.
- Run epochs $j = 0, 1, 2, \ldots$.
- In epoch j, we assume $L_t(i^*) \leq 2^j$, and set ϵ to minimize

  $$\frac{\log K}{\epsilon} + \epsilon 2^j,$$

  the minimizing ϵ being $\epsilon_j = \sqrt{\frac{2 \ln K}{2^j}}$. Epoch j ends when the assumption $L_t(i^*) \leq 2^j$ is violated.
- Then, when epoch j ends, we reboot the HEDGE algorithm and start all over in epoch $j+1$.

The regret occured in epoch j is

$$\text{Regret(epoch } j) \leq \frac{\ln K}{\epsilon_j} + \epsilon_j 2^j \leq \sqrt{2\ln(K)2^j}.$$

If we end in epoch J, our regret is

$$\text{REGRET} \leq \sum_{j=0}^{J} \sqrt{2\ln(K)2^j} \leq \sqrt{2\ln K} \sum_{j=0}^{J} 2^{j/2} = O(\sqrt{2\ln K 2^J}) = O(\sqrt{2\ln(K)L_T(i^*)}),$$

where we get the last equality by observing that if we end in epoch J, the assumption that the best expert's loss $L_t(i^*) \leq 2^{J-1}$ must have been violated in epoch $J-1$, so $L_T(i^*) > 2^{J-1}$.