4810-1183 Approximation and Online Algorithms with Applications
Lecture Note 9: Online Machine Learning Algorithm

Problem Definition

We are working on stock day trade and we are focusing on a particular stock GOOG. On every day in the morning, we will make a decision to buy or not to buy the stock, but, if we buy it, we will sale it in the evening. If we buy and the stock price gets higher on the day, we will receive a profit.

Suppose that we do this day trade repetitively for T days. We receive opinions from n stock experts in the morning of every day. The experts advise us to buy and not to buy GOOG on each day. The opinions on day t are denoted by $x_1^{(t)}, x_2^{(t)}, \dots, x_n^{(t)} \in \{\text{buy}, \text{no buy}\}$. We say that expert i give us a correct opinion on day t, if $x_i^{(t)} = \text{buy}$ and the stock price go up on the day, or if $x_i^{(t)} = \text{no} - \text{buy}$ and the stock price go down on day t. Otherwise, we will say that the expert give us an incorrect opinion.

The opinions can be correct or incorrect based on how good each expert knows about GOOG, but we do not know who the best expert is on day 1. However, based on the experts' performances on earlier day, we want to find a set of good experts and follow them.

The problem definition can be stated as follows:

<u>Input</u>: For all $t \in \{1, \dots, T\}$, $x_1^{(t)}, x_2^{(t)}, \dots, x_n^{(t)} \in \{\text{buy}, \text{no buy}\}$, $r^{(t)} \in \{\text{up}, \text{down}\}$

<u>Output</u>: Our decision $d^{(t)} \in \{\text{buy}, \text{no buy}\}$

<u>Constraint</u>: $d^{(t')}$ is calculated only for $x_1^{(t)}, \dots., x_n^{(t)}$ for $t \le t'$ and $r^{(t)}$ for $t < t'$

<u>Objective Function</u>: Denote $p^{(t)} = \begin{cases} 1 & \text{if } d^{(t)} = \text{buy } and\ r^{(t)} = up, \\ 1 & \text{if } d^{(t)} = \text{no buy } and\ r^{(t)} = down \cdot \\ 0 & \text{otherwise} \end{cases}$

Maximize $\sum_t p^{(t)}$

It is important to note that to make a decision at time t, denoted by $d^{(t)}$, we will know the expert advices $x_1^{(t)}, x_2^{(t)}, \dots, x_n^{(t)}$ but not the result $r^{(t)}$. If we know $r^{(t)}$, this problem would be very easy.

Algorithm

The algorithm for the online problem, called as halving algorithm, is as follows:

```
1: For all $1 \le i \le n$, $w_i = 1$.
2: For $t = 1$ to $T$:
3:     If $\sum_{i:x_i^{(t)}=\text{buy}} w_i \ge \sum_{i:x_i^{(t)}=\text{no buy}} w_i$:
4:          $r^{(t)} = \text{buy}$
5:     Else:
6:          $r^{(t)} = \text{no buy}$
7:     For all expert $i$ that makes a wrong suggestion, $w_i \leftarrow w_i/2$
```

<u>The weight w_i shows how much we believe in the expert i</u>. At the beginning, we set the weight for all experts to 1, which indicates that we believe in all experts equally. Then, in the evening of every days, we will check if a particular experts make a wrong suggestion, and reduce the weight of those who make wrong suggestion by half if Line 7 of the algorithm.

We make a decision $r^{(t)}$ based on the condition in Line 3. There, we calculate the sum of weights for experts who suggest buying the stock, compare it to the sum of weights for experts who do not suggest, and we buy the stock when the first weight sum is larger than the second.

Example

Suppose that $n = 3$ and $T = 3$. The halving algorithm works as in the following table:

Time	Expert 1	Expert 2	Expert 3	Our Decision
1	Suggestion: Buy $w_1 = 1$	Suggestion: No Buy $w_2 = 1$	Suggestion: No Buy $w_3 = 1$	Sum weight for buy = 1 Sum Weight for no buy = 1 + 1 = 2 Decision: no buy **Wrong Decision**
2	Suggestion: No Buy $w_1 = 1$	Suggestion: Buy $w_2 = 0.5$	Suggestion: No Buy $w_3 = 0.5$	Sum weight for buy = 0.5 Sum Weight for no buy = 1 + 0.5 = 1.5 Decision: no buy **Right Decision**
3	Suggestion: Buy $w_1 = 1$	Suggestion: No Buy $w_2 = 0.5$	Suggestion: No Buy $w_3 = 0.25$	Sum weight for buy = 1 Sum weight for no buy = 0.5 + 0.25 = 0.75 Decision: buy **Right Decision**

Theory

We can prove the following theory for the algorithm.

Theorem 1: Let m be the number of mistakes by the best expert, i.e.

$$m := \min_i(\text{number of mistakes by expert } i).$$

Then, the number of incorrect decisions obtained by the algorithm is at most $2.41(m + \lg_2 n)$.

Proof: As a weight of an expert is reduced by half after one mistake and the best expert makes a mistake for m times, we know that, at the end of all investments, the weight of the best person is $1/2^m$.

Denote the sum of all of the expert weights $\sum_i w_i$ after the t^{th} iteration as W_t. We know that $W_T \geq 1/2^m$ because the weight sum must include the value is a weight of the best expert after the T^{th} iteration, which is $1/2^m$.

Next, let us find a relationship of W_t and W_{t+1} when we make an incorrect decision. We know that

$W_t =$ (sum of weight with correct suggestion C_t) + (sum of weight with incorrect suggestion I_t).

We will make an incorrect decision when $I_t \geq C_t$, which means that $I_t \geq W_t/2$. At Step 7 of the algorithm, we will reduce the weight of all experts who make an incorrect suggestion by half. That results in reducing the sum of all the weights I_t by half. Hence, we will have

$$W_{t+1} = C_t + \frac{I_t}{2} = (C_t + I_t) - \frac{I_t}{2} = W_t - \frac{I_t}{2} \leq W_t - (\frac{W_t}{2})/2 \ = \frac{3}{4} W_t.$$

The weight sum W_t will be cut by at least 25% for each incorrect decision. As all $w_i = 1$ at the beginning of the algorithm, the weight sum at the beginning W_0 is n. Suppose that we make M incorrect decisions. That weight sum will be reduced by at least 25% for at most M times. At the end of the algorithm, the weight sum $W_T \leq \left(\frac{3}{4}\right)^M n$. As discussed at the beginning of this proof, we have $W_T \geq 1/2^m$. We know that

$$\left(\frac{3}{4}\right)^M n \geq W_t \geq \frac{1}{2^m}$$

$$\lg_2 \left[\left(\frac{3}{4}\right)^M n\right] \geq \lg_2 \left(\frac{1}{2^m}\right)$$

$$\lg_2 \left[\left(\frac{3}{4}\right)^M\right] + \lg_2 n \geq -m$$

$$M \lg_2 0.75 + \lg_2 n \geq -m$$

$$-0.415M + \lg_2 n \geq -m$$

$$m + \lg_2 n \geq 0.415M$$

$$M \leq 2.41(m + \lg_2 n)$$

❑

The best expert in our example is Expert 1, who does not make any mistake in her suggestion. We have $m = 0$. Then, $2.41(m + \lg_2 n) = 2.41(0 + \lg_2 3) = 3.82$. As we make one incorrect decision in the example, the number of incorrect decisions is smaller than the upper bound given in the theorem.

Using Randomization

We have ever used a randomized algorithm in the randomized rounding scheme earlier in this course. In this section, we introduce that randomized algorithm to give a good performance in the online problem.

Let us consider the following algorithm:

```
1: For all $1 \leq i \leq n$, $w_i = 1$.
2: For $t = 1$ to $T$:
3:     $W_{\text{buy}} \leftarrow \sum_{i:x_i^{(t)}=\text{buy}} w_i$:
4:     $W_{\text{no buy}} \leftarrow \sum_{i:x_i^{(t)}=\text{no buy}} w_i$
5:     $d^{(t)} \leftarrow$ buy with probability $P_{\text{buy}} \coloneqq W_{\text{buy}}/(W_{\text{buy}} + W_{\text{no buy}})$
       $d^{(t)} \leftarrow$ no buy with probability $1 - P_{\text{buy}}$
6:     For all expert $i$ that makes a wrong suggestion, $w_i \leftarrow w_i/2$
```

Instead of always choosing the choice with a larger weight sum, we will choose to buy the stock with a probability depending the two weight sums. If W_{buy} is much larger than $W_{\text{no buy}}$, we will likely to buy the stock. On the other hand, we will not likely to buy if $W_{\text{no buy}}$ is much larger than W_{buy}.

We will have the following theorem for the randomized algorithm.

Theorem 2: The expected number of incorrect decisions is no larger than $1.39m + 2\ln n$.

Proof: Let $F^{(t)}$ be a probability that we make an incorrect decision at time t. The expected number of incorrect decisions is then equal to $\sum_t F^{(t)}$.

We will again consider a sum of all weights $W^{(t)}$. Recall that $W^{(0)} = n$ and $W^{(t)} = C^{(t)} + I^{(t)}$ when $C^{(t)}$ is the weight sum of experts who give a correct suggestion and $I^{(t)}$ is the sum of the weight of experts who give an incorrect suggestion. By that, we have $F^{(t)} = \frac{I^{(t)}}{C^{(t)}+I^{(t)}} = \frac{I^{(t)}}{W^{(t)}}$ and

$$W^{(t+1)} = C^{(t)} + \frac{I^{(t)}}{2} = W^{(t)} - \frac{I^{(t)}}{2} = W^{(t)}\left[1 - \frac{1}{2}\frac{I^{(t)}}{W^{(t)}}\right] = W^{(t)}\left[1 - \frac{F^{(t)}}{2}\right].$$

We then have

$$W^{(T)} = W^{(0)}\prod_t\left[1 - \frac{F^{(t)}}{2}\right] = n\prod_t\left[1 - \frac{F^{(t)}}{2}\right].$$

We know from the previous proof that $W^{(T)} \geq 1/2^m$. Hence,

$$n\prod_t\left[1 - \frac{F^{(t)}}{2}\right] \geq \frac{1}{2^m}.$$

As $1 - x \leq e^{-x}$, we have

$$\frac{1}{2^m} \leq n\prod_t\left[1 - \frac{F^{(t)}}{2}\right] \leq n \cdot e^{-\frac{1}{2}\sum_t F^{(t)}}$$

$$\ln 2^{-m} \leq \ln\left(n \cdot e^{-\frac{1}{2}\sum_t F^{(t)}}\right)$$

$$-m\ln 2 \leq \ln n + \ln e^{-\frac{1}{2}\sum_t F^{(t)}}$$

$$-0.693m \leq \ln n - \frac{1}{2}\sum_t F^{(t)}$$

$$\frac{1}{2}\sum_t F^{(t)} \leq 0.693m + \ln n$$

$$\sum_t F^{(t)} \leq 1.39m + 2\ln n.$$

❑

The bound in Theorem 2, $1.39m + 2\ln n$, is smaller than the bound in Theorem 1, $2.41m + 2.41\lg_2 n$. However, we cannot say from here that the randomized algorithm has a better performance than the first algorithm. Theorem 2 is a bound for the expected value, which is an average performance after we run the algorithm for 1,000,000 times. In the worst case, we might get a performance that is worse than that from Algorithm 1.

Crowdsourcing

Let us consider the application of the above algorithm on crowdsourcing. In any matching learning, we always want to find a function f that can capture a thing that we want to learn about. To find a good function f, we have to give a large number of inputs and outputs to train machine learning algorithms. For example, we may want to find a function f that takes an image an input and gives an answer if the image contains cancer cells.

It is usually not very hard to find training inputs as there are a lot of images from patience. More hospitals and government units are publishing the data. However, it is not very easy to find training outputs as that information is not always published. We may have to find some medical doctors who are good in cancer analysis. The cost for that may be too large to effort. Instead of one expert who are expensive, it might be more reasonable to have a lot of people voting for a correct answer. That is called as **crowdsourcing**.

It is always a problem how to motivate people who give us outputs, called as workers, to give us a precise answer as much as we can. If we give the same incentive for all outputs given, workers might give us random outputs. Because of that, it is very important to have a way to evaluate each worker, believe workers who know give us precise outputs, and give less incentive to workers who give us random outputs.

We can solve that using the algorithms provided in this lecture note. We will have experts provide gold standard for a very few outputs. Then, we randomly insert inputs with outputs from experts. Without informing workers which inputs we have the gold standard outputs, we will have the workers give us all the outputs, compare them with the gold standard, and evaluate each worker by the comparison results.

We can use the weighted majority voting (Line 3 of the first algorithm) to combine outputs suggested from workers. Also, we can reduce the weight of workers that give us an incorrect information by half as in Line 7 of the first algorithm. The payment for each worker can depend on his/her weight. Although we still do not have a good theoretical analysis on this idea, it is shown in [2] that it gives a fair performance.

Reference

[1] A. Blum, "*On-Line Algorithms in Machine Learning*", Chapter 14 of "Online Algorithms: the state of the art", Lecture Note in Computer Science Vol. 1442, Fiat and Woeginger eds., 1998.

[2] Z. Hu, Y. Liang, J. Zhang, Z. Li, Y. Liu, "*Inference Aided Reinforcement Learning for Incentive Mechanism Design in Crowdsourcing*", Proceeding of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), 2018.