The multiplicative weight algorithms

Huy L. Nguyễn

This time we will take an opposite approach to uncertainty and do not assume any knowledge about future outcomes. One reason for the lack of knowledge is the lack of resources: we might not have the resources to consider all possible future outcomes, learn the probability distributions, and maximize the expected utility.

1 Weighted majority

We will illustrate the algorithm with a concrete example. Assume that we are interested in the price of one particular stock. Everyday, the stock can either move up or down. Each morning, we would like to predict the stock movement for that day. Since we do not know anything about the stock, we can watch n different experts giving their predictions. The goal is to minimize the number of mistakes.

Question. What would be a good performance? Can we aim to make fewer than, say 40% mistakes?

Our target is to do as well as possible compared with the best expert in hindsight.

One idea is to follow the majority vote of the experts. However, this does not work because it is possible that most experts are consistently wrong on most days and just one expert is correct on all days. If we follow the majority, we will also be wrong most of the times.

The second idea is to follow the best expert so far (in case of tie, we will follow the one with the minimum index). Consider the following simple example. The stock moves up every day. There are 3 experts. Expert 1 predicts the stock to go down on days $2, 5, 8, 11, \ldots$ Expert 2 predicts the stock to go down on days $3, 6, 9, 12, \ldots$ Expert 3 predicts the stock to go down on days $4, 7, 10, \ldots$. On day 2, we follow expert 1 and guess wrong. On day 3, we follow expert 2 and guess wrong. On day 4, we follow expert 1 and guess wrong. The cycle then repeats. We are always wrong but the experts are correct roughly $2/3$ of the times. By increasing the number of experts, we are always wrong but the experts are wrong roughly T/n of the times.

The third idea is to reweight the experts gradually as we learn more about the performance in the past. We consider the following algorithm:

Theorem 1.1. *Let $m_i^{(t)}$ be the number of mistakes of expert i after t steps. Let $M^{(t)}$ be the number of mistakes our algorithm makes after t steps. We have the following bound for every i:*

$$M^{(T)} \leq 2(1+\varepsilon)m_i^{(T)} + \frac{2\ln n}{\varepsilon}$$

Proof. Consider the potential function $\Phi^{(t)} = \sum_{i=1}^{n} w_i^{(t)}$. We will show that

- when we are wrong, the potential decreases.
- if there is a good expert, the potential cannot be too small.

Algorithm 1 Weighted majority.

1: Initialize weights $w_i^{(0)} = 1 \ \forall i \in \{1, 2, \ldots, n\}$ and a constant $\varepsilon \in (0, 1/2)$.
2: **for** t from 1 to T **do**
3: Make the prediction by the majority vote weighted by the weights $w_i^{(t-1)}$
4: After the result is revealed, set

$$a_i^{(t)} = \begin{cases} 1 \text{ if } i \text{ is wrong} \\ 0 \text{ otherwise} \end{cases}$$

5: Update $w_i^{(t)} \leftarrow w_i^{(t-1)}(1 - \varepsilon \cdot a_i^{(t)})$
6: **end for**

- conclude that we cannot be wrong too often.

Consider an iteration t where we are wrong. It must be the case that the weighted majority of the experts are wrong. Thus, at least half of the weight decreases by a factor $1-\varepsilon$. Thus, the total weight decreases by at least a factor $1 - \varepsilon/2$:

$$\Phi^{(t)} \leq \Phi^{(t-1)}(\frac{1}{2} + \frac{1-\varepsilon}{2}) = \Phi^{(t-1)}(1 - \varepsilon/2)$$

Thus, by induction, we get $\Phi^{(T)} \leq n(1 - \varepsilon/2)^{M^{(T)}}$.

For the second step, observe that $\Phi^{(T)} \geq w_i^{(T)} = (1-\varepsilon)^{m_i^{(T)}}$. This is because $w_i^{(0)} = 1$ and it is decreases by a factor $1 - \varepsilon$ every time expert i is wrong. Combining the two inequalities, we obtain

$$\begin{aligned} (1-\varepsilon)^{m_i^{(T)}} &\leq n(1-\varepsilon/2)^{M^{(T)}} \\ m_i^{(T)} \ln(1-\varepsilon) &\leq \ln n + M^{(T)} \ln(1-\varepsilon/2) \\ m_i^{(T)} \frac{-\ln(1-\varepsilon)}{-\ln(1-\varepsilon/2)} + \frac{\ln n}{-\ln(1-\varepsilon/2)} &\geq M^{(T)} \end{aligned}$$

Observe that $-x - x^2 \leq \ln(1-x) \leq -x \ \forall x \in (0, 1/2]$ so $-\ln(1-\varepsilon) \leq \varepsilon + \varepsilon^2$ and $-\ln(1-\varepsilon/2) \geq \varepsilon/2$. Substituting these facts into the last line above, we obtain the theorem. □

2 Randomized weighted majority

In this section, we will develop a new algorithm that avoids the factor 2 in the number of mistakes we suffer compared with the best expert. This is because we do not consider how strong the majority vote is (it could be 99 vs 1 or 51 vs 49). Instead, we will use a randomized strategy that follow the majority vote with probability proportional to its weight. Note that this can also be interpreted as following a random expert with probability proportional to his weight.

Theorem 2.1. *Let $m_i^{(t)}$ be the number of mistakes of expert i after t steps. Let $M^{(t)}$ be the* expected *number of mistakes our algorithm makes after t steps. We have the following bound for every i:*

$$M^{(T)} \leq (1+\varepsilon) m_i^{(T)} + \frac{\ln n}{\varepsilon}$$

Algorithm 2 Randomized weighted majority.

1: Initialize weights $w_i^{(0)} = 1 \ \forall i \in \{1, 2, \ldots, n\}$ and a constant $\varepsilon \in (0, 1/2)$.
2: **for** t from 1 to T **do**
3: Pick expert i with probability $\frac{w_i^{(t-1)}}{\Phi^{(t-1)}}$ and follow his prediction.
4: After the result is revealed, set

$$a_i^{(t)} = \begin{cases} 1 \text{ if } i \text{ is wrong} \\ 0 \text{ otherwise} \end{cases}$$

5: Update $w_i^{(t)} \leftarrow w_i^{(t-1)}(1 - \varepsilon \cdot a_i^{(t)})$
6: **end for**

Proof. On day t the probability that we make mistake is

$$b^{(t)} = \sum_{i=1}^{n} Pr[\text{we follow expert } i \text{ and } i \text{ is wrong}] = \sum_{i=1}^{n} \frac{w_i^{(t-1)} a_i^{(t)}}{\Phi^{(t-1)}}$$

We have

$$\begin{aligned} \Phi^{(t)} &= \sum_{i=1}^{n} w_i^{(t-1)}(1 - \varepsilon a_i^{(t)}) \\ &= \sum_{i=1}^{n} w_i^{(t-1)} - \varepsilon \sum_{i=1}^{n} w_i^{(t-1)} a_i^{(t)} \\ &= \Phi^{(t-1)}(1 - \varepsilon b^{(t)}) \\ &\leq \Phi^{(t-1)} \exp(-\varepsilon b^{(t)}) \end{aligned}$$

Thus, by induction, we get $\Phi^{(T)} \leq n \exp(-\varepsilon M^{(T)})$.

Observe that $\Phi^{(T)} \geq w_i^{(T)} = (1-\varepsilon)^{m_i^{(T)}}$. This is because $w_i^{(0)} = 1$ and it is decreases by a factor $1-\varepsilon$ every time expert i is wrong. Combining the two inequalities and taking the log of both sides, we obtain

$$\begin{aligned} (1-\varepsilon)^{m_i^{(T)}} &\leq n \exp(-\varepsilon^{M^{(T)}}) \\ m_i^{(T)} \ln(1-\varepsilon) &\leq \ln n - \varepsilon M^{(T)} \\ M^{(T)} &\leq m_i^{(T)} \frac{-\ln(1-\varepsilon)}{\varepsilon} + \frac{\ln n}{\varepsilon} \end{aligned}$$

Observe that $-\ln(1-\varepsilon) \leq \varepsilon + \varepsilon^2$. Substituting this fact into the last line above, we obtain the theorem. □

3 The multiplicative weight algorithm

Finally we slightly generalize the setting. Each day, we can pick an expert and imitate his action. The action of expert i on day t will suffer a cost $a_i^{(t)}$ in the range $[-1, 1]$ (negative cost can be viewed as reward). The goal is to minimize the total cost.

We can observe that the exact same proof goes through and we obtain the following theorem.

Algorithm 3 The multiplicative weight algorithm.

1: Initialize weights $w_i^{(0)} = 1 \ \forall i \in \{1, 2, \ldots, n\}$ and a constant $\varepsilon \in (0, 1/2)$.
2: **for** t from 1 to T **do**
3: Pick expert i with probability $\frac{w_i^{(t-1)}}{\Phi^{(t-1)}}$ and follow his action.
4: Action of expert i suffers cost $a_i^{(t)}$.
5: Update $w_i^{(t)} \leftarrow w_i^{(t-1)}(1 - \varepsilon \cdot a_i^{(t)})$
6: **end for**

Theorem 3.1. *Let $m_i^{(t)}$ be the total cost of expert i after t steps. Let $M^{(t)}$ be the* expected *cost of our algorithm after t steps. We have the following bound for every i:*

$$M^{(T)} \leq m_i^{(T)} + \varepsilon \sum_{t=1}^{T} |a_i^{(t)}| + \frac{\ln n}{\varepsilon}$$

4 Solving the set cover linear program

The multiplicative weight algorithm can be used to solve LP. In this section, we will describe its application to one specific LP we encountered before: the set cover LP. Recall that we have m sets $S_1, S_2, \ldots, S_m$ over a universe of n elements. We would like to pick the minimum number of sets that cover all the elements.

$$\begin{aligned} \min \sum_{i=1}^{m} x_i \\ \sum_{i:j \in S_i} x_i \geq 1 \ \forall j \\ x_i \geq 0 \ \forall i \end{aligned}$$

We can think of each constraint as an expert. Initially, we have equal weights on the constraints $w_j^{(0)} = 1$. In round t, adding up all the constraints with the weights gives us a single constraint

$$\sum_{j=1}^{m} w_j^{(t-1)} \left(\sum_{i:j \in S_i} x_i \right) \geq \sum_{j=1}^{m} w_j^{(t-1)}$$

While solving LP with many constraint is hard, solving LP with just one constraint is much easier. Suppose we can solve the following single-constraint LP and obtain the solution $x^{(t)}$.

$$\begin{aligned} \min \sum_{i=1}^{m} x_i \\ \sum_{i=1}^{m} \left(\sum_{j \in S_i} w_j^{(t-1)} \right) x_i \geq \sum_{j=1}^{m} w_j^{(t-1)} \\ x_i \geq 0 \ \forall i \end{aligned}$$

Notice that the solution is to set exactly one coordinate of x to non-zero (which one?) and the rest to zero.

Note that any solution for the original LP is also a solution here. Thus, if OPT is the optimal value for the original LP then the solution $x^{(t)}$ for the problem in round t has $\sum_{i=1}^{m} x_i^{(t)} \leq OPT$.

We have the constraints as experts so we need to assign the costs for their actions. The cost for expert i is defined as how much the solution $x^{(t)}$ "overflows" constraint i.

$$a_i = \frac{1}{OPT}\left(\sum_{i:j\in S_i} x_i^{(t)}\right)$$

Note that $a_i \in [0,1]$. What is the meaning of the reweighting? when a set is over-covered, we lower its weight so that we don't try to cover it as much next time.

Question. What is our expected cost in round t?

It turns out to be exactly $\frac{1}{OPT}$ times the ratio between the LHS and the RHS of the constraint, which is at least $\frac{1}{OPT}$.

We apply multiplicative weights with $T = \frac{4OPT^2 \ln n}{\gamma^2}$ iterations and $\varepsilon = \frac{\gamma}{2OPT}$. At the end, we will consider the average solution

$$\bar{x} = \frac{1}{T}\sum_{i=1}^{T} x^{(t)}$$

Because each $x^{(t)}$ has small objective value, the average also has small objective value. The question is, does it satisfy the constraints?

To reason about this, we look at the guarantee of the multiplicative weight algorithm. Our cost per round is at least $1/OPT$ so the total cost is at least T/OPT. We now compare our cost with the cost of expert j:

$$\frac{T}{OPT} \leq \frac{1}{OPT}\left(T\sum_{i:j\in S_i} \bar{x}_i\right) + \varepsilon T + \frac{\ln n}{\varepsilon}$$

Substituting in the value of T, ε, we get

$$\sum_{i:j\in S_i} \bar{x}_i \geq 1 - \gamma$$

Thus, our solution is nearly-feasible and has optimal objective value. We can also obtain a solution that is feasible and near-optimal objective value by scaling the solution up by a factor $\frac{1}{1-\gamma}$.