Multiplicative-Weight Update Method

Anil Maheshwari

anil@scs.carleton.ca
School of Computer Science
Carleton University
Canada

Outline

Problem

Expert Investor Without an Expertise

Model:

1. Access to n experts (newspapers, stock briefs, . . .)
2. Predict whether TSX will go $\uparrow$ or $\downarrow$ at the end of each day.
3. Reward of $\$0$/per day for correct prediction.
4. Costs us $\$1$/per day for wrong prediction.

Problem: Devise an algorithm that makes prediction for each day. Suppose we are at day t, where $t \in \{1, \ldots, T\}$. Algorithm can use previous predictions $+$ experts advises for days $1, \ldots, t-1$

Objective: At the end of T days we want to be competitive with respect to the best expert. Our cost (loss) is not significantly higher than the cost of any expert (including the best expert).

Warmup

Two Scenarios

We will consider the following scenarios:

1. There is at least one (unknown) expert who is always correct.
2. The best expert (unknown) makes $\leq m$ mistakes over the period of T days.

≥ 1 Real Experts

Extra Knowledge: Among all the n experts there is at least one (unknown) expert who is always correct.

Let the set of experts be $E = \{1, \ldots, n\}$

For each day $t := 1$ to T do:

Step 1: Among all the remaining experts in E, poll them to find the prediction of the majority of them for that day. Record that as the prediction of the algorithm.

Step 2: Observe the true outcome at the end of the day. Discard all those experts that predicted wrong from E from future considerations.

Warmup (contd.)

#Wrong Predictions

Algorithm makes at most $O(\log n)$ wrong predictions

Multiplicative Weight Update Method - Idea

Extra Knowledge: The best expert (unknown) makes $\leq m$ mistakes over the period of T days.

Let the set of experts be $E = \{1, \ldots, n\}$
For each expert i, set its weight $w_i^1 = 1$

For each day $t := 1$ to T do:

Step 1: Find the weighted majority prediction of the experts. Sum total the weights of all the experts that predict $\uparrow$ (respectively, $\downarrow$). Whichever of the two sums is higher is the prediction for day t.

Step 2: Observe the true outcome at the end of the day t

Step 3: For all the experts i that predicted wrongly, their weight is set to $w_i^{t+1} = w_i^t/2$. For all others, $w_i^{t+1} = w_i^t$.

Potential Function

Potential Function

Potential Function Φ^t

Define the potential function Φ^t for day $t \in \{1, \ldots, T\}$ to be the sum total of the weights of all the experts at the start of day t, i.e. $\Phi^t = \sum_{i=1}^{n} w_i^t$

Observations:

1. $\Phi^1 = n$
2. If the algorithm makes a wrong prediction on day t, $\Phi^{t+1} \leq \frac{3}{4}\Phi^t$
3. If the algorithm has made M mistakes in T days, its total weight at the end of day T, $\Phi^{T+1} \leq \left(\frac{3}{4}\right)^M \Phi^1 = \left(\frac{3}{4}\right)^M n$

Bounding Potential Function

Assume that the best Expert is i (we don't know it's identity).

Bounds on Potential Function

Weight of the expert i at the end of day T is $\geq (\frac{1}{2})^m$. Therefore, $\left(\frac{1}{2}\right)^m \leq \Phi^{T+1} \leq \left(\frac{3}{4}\right)^M n$

Number of mistakes

Bounding number of mistakes

The algorithms makes at most $2.41(m + \log n)$ mistakes.

Proof: We had $\left(\frac{1}{2}\right)^m \leq \Phi^{T+1} \leq \left(\frac{3}{4}\right)^M n$. Take $\log$'s:

$$\begin{aligned} -m &\leq M \log\left(\frac{3}{4}\right) + \log n \\ -M \log\left(\frac{3}{4}\right) &\leq m + \log n \\ M \log\left(\frac{4}{3}\right) &\leq m + \log n \\ M &\leq 2.41(m + \log n) \end{aligned}$$

□

Replacing $\frac{1}{2}$

Choose $\eta \in (0, \frac{1}{2}]$

Let the set of experts be $E = \{1, \ldots, n\}$
For each expert i, set its starting weight $w_i^1 = 1$

For each day $t := 1$ to T do:

Step 1: Find the weighted majority prediction of the experts. Sum total the weights of all the experts that predict $\uparrow$ (respectively, $\downarrow$). Whichever of the two sums is higher is the prediction for day t.

Step 2: Observe the true outcome at the end of the day t

Step 3: For all the experts i that predicted wrongly, their weight is set to $w_i^{t+1} = (1 - \eta) w_i^t$. For all others, $w_i^{t+1} = w_i^t$.

Analysis

New Bound

For any $\eta \in (0, \frac{1}{2}]$, we have $M \leq 2(1+\eta)m + \frac{2}{\eta}\log n$

Proof: Potential function equation: $(1-\eta)^m \leq \Phi^{T+1} \leq (1-\frac{\eta}{2})^M n$

Take $\log$'s and use $-\eta - \eta^2 \leq \ln(1-\eta) \leq -\eta$ for $\eta \in [0, \frac{1}{2}]$:

$$\begin{aligned} m\ln(1-\eta) &\leq M\ln(1-\frac{\eta}{2}) + \ln n \\ -m(\eta+\eta^2) &\leq -M\frac{\eta}{2} + \ln n \\ M\frac{\eta}{2} &\leq \ln n + (\eta+\eta^2)m \\ M &\leq \frac{2}{\eta}\ln n + 2(1+\eta)m \end{aligned}$$

□

Randomization

An Example

For any $\eta \in (0, \frac{1}{2}]$, we have $M \leq 2(1+\eta)m + \frac{2}{\eta}\log n$

Example: Two experts A and B.
A predicts $(\uparrow, \downarrow)$ correctly only on even numbered days
B predicts $(\uparrow, \downarrow)$ correctly only on odd numbered days

Best Expert: Wrong half the times.

How many times the Algorithm predicts wrongly?

Improvements (contd.)

How to remove the multiplicative factor of 2 in $M \leq \mathbf{2}(1+\eta)m + \frac{2}{\eta}\log n$?

Use Randomization

Objectives

Loss/Cost: Assume that the loss (costs) are real numbers in the interval $[0, 1]$.

Let $m_i^t \in [0, 1]$ denote the loss of expert $i \in \{1, \ldots, n\}$ on day $t \in \{1, \ldots, T\}$

Objective: Algorithm to be competitive against the cost incurred by the best expert. Assume i is the (unknown) best expert.

$M^t =$ Expected cost that the algorithm incurs on day t.

Our algorithm should have the following guarantee:

$$\sum_{i=1}^{T} M^t \leq \frac{\ln n}{\eta} + (1 + \eta) \sum_{i=1}^{T} m_i^t$$

Total expected cost over T days is within an additive factor $\frac{\ln n}{\eta}$ and a multiplicative factor $(1 + \eta)$ of the best expert.

Randomized MWU

Multiplicative Weight Update Method

Set of experts $E = \{1, \ldots, n\}$.
Let η be any real number in $[0, \frac{1}{2}]$
For each expert i, set its initial weight $w_i^1 = 1$

For each day $t := 1$ to T do:

Step 1: Define $\Phi^t = \sum_{i=1}^{n} w_i^t$

For each expert i, compute $p_i^t = \frac{w_i^t}{\Phi^t}$

Step 2: Choose an expert based on their probabilities and predict $(\uparrow, \downarrow)$ according to the chosen expert

Step 3: Update Weights: For each expert i set $w_i^{t+1} = w_i^t(1 - \eta m_i^t)$

± 1 Costs

MWU with costs in $[-1, 1]$

The costs of each expert can be positive or negative, i.e. $m_i^t \in [-1, 1]$. Use the same algorithm as for the cost $[0, 1]$:

MWU with costs in $[-1, 1]$

Set of experts $E = \{1, \ldots, n\}$.
Let η be any real number in $[0, \frac{1}{2}]$
For each expert i, set its initial weight $w_i^1 = 1$

For each day $t := 1$ to T do:

Step 1: Define $\Phi^t = \sum_{i=1}^{n} w_i^t$

For each expert i, compute $p_i^t = \frac{w_i^t}{\Phi^t}$

Step 2: Choose an expert based on their probabilities and predict $(\uparrow, \downarrow)$ according to the chosen expert

Step 3: Update Weights: For each expert i set $w_i^{t+1} = w_i^t(1 - \eta m_i^t)$

Result

MWU with ± 1-costs

$$\sum_{t=1}^{T} M^t \leq \frac{\ln n}{\eta} + \eta T + \sum_{t=1}^{T} m_i^t$$

Cost of MWU

By setting $\eta = \sqrt{\frac{\ln n}{T}}$ in $\sum_{t=1}^{T} M^t \leq \frac{\ln n}{\eta} + \eta T + \sum_{t=1}^{T} m_i^t$, we obtain

$$\sum_{t=1}^{T} M^t \leq 2\sqrt{T \ln n} + \sum_{t=1}^{T} m_i^t$$

Interpretation of the Result

$$\sum_{t=1}^{T} M^t \leq 2\sqrt{T \ln n} + \sum_{t=1}^{T} m_i^t$$

Cost of MWU algorithm is off by an additive factor that is proportional to the square root of the product of the number of days and the number of experts as compared to the best expert.

Average Error: Consider the average error on each day (divide by T):

$$\frac{1}{T}\sum_{t=1}^{T} M^t \leq 2\sqrt{\frac{\ln n}{T}} + \frac{1}{T}\sum_{t=1}^{T} m_i^t$$

Observe that as T increases the average error drops down. Therefore, MWU method is able to learn from experts reasonably well when executed over a number of days.

Reference

- Arora, Hazan and Kale, The multiplicative weights update method: a meta-algorithm and applications, Theory of Computing 8(1): 121-164, 2012.

- Several Lecture Notes