Lecture 10: *Online Learning for Experts Problem & Machine Learning*

Advanced Algorithms

Sid Chi-Kin Chau

Australian National University

sid.chau@anu.edu.au

October 5, 2022

Experts' Predictions are always *Inconsistent*!

"At least we are consistently inconsistent."

≪How to aggregate experts' inconsistent opinions?≫

Experts for Stock Market Predictions

- You listen to n experts for investment predictions in stock markets. Every day, each of them predicts whether the stock will go up or down

Day	Expert 1 👩‍💼	Expert 2 👨🏻‍💼	Expert 3 👨🏽‍💼	Actual
1	📈	📉	📈	📈
2	📉	📈	📈	📉
3	📉	📈	📉	📉
4	📈	📈	📈	📉

- Experts' predictions may be wrong sometimes
- Goal: Pick the strategy to do as well as the best expert
- What is our strategy?
  - Benchmark the accuracy of experts' past predictions
  - Aggregate the predictions based on the their past accuracy

Experts Problem

Definition (Experts Problem)

- For $t = 1, ..., T$ (days on the stock market), each expert $i = 1, \ldots, n$ predicts "yes" or "no"
- Aggregator decide either yes or no based on individual experts' predictions
- Adversary, with knowledge of Aggregator's decision and experts' predictions, makes the actual yes-or-no outcome
- Aggregator observes the actual outcome, and suffers a cost if his decision is incorrect

- Aggregator's role is to make as few mistakes as possible
  - But since the experts may be unhelpful and the outcomes can be wrong, Aggregator can only hope for a comparable performance to the best expert, in hindsight
- The number of mistakes in excess of the best expert's mistakes is called *regret*
- Adversary's role is not to make Aggregator be wrong all the time (Adversary is omniscient who can easily make the opposite outcome of what Aggregator decided)
  - Nonetheless, Adversary wants to inflict as much regret as possible on Aggregator

Weighted Majority Algorithm

Weighted Majority Algorithm (WMA)

- Assign a weight $w_i^{(1)} = 1$ to each expert i
- On each t-th day, Aggregator decides yes or no based on a majority vote of all experts, weighted by $(w_i^{(t)}, \ldots, w_n^{(t)})$: if $\sum_{i:i \text{ says yes}} w_i^{(t)} > \sum_{i:i \text{ says no}} w_i^{(t)}$, then yes, otherwise, no
- After observing the outcome, for every incorrect expert i, set $w_i^{(t+1)} \leftarrow w_i^{(t)}/2$

Theorem

Let $M_{\mathsf{WMA}}^{(t)}$ and $M_i^{(t)}$ be the number of mistakes that WMA and expert i make, respectively, until time t. For any sequence of outcomes, any duration T and any expert i:

$$M_{\mathsf{WMA}}^{(T)} \leq 2.41(M_i^{(T)} + \log n), \quad \text{Regret} = M_{\mathsf{WMA}}^{(T)} - \min_{i=1,\ldots,n} M_i^{(T)} \leq 1.41(M_i^{(t)} + \log n)$$

Weighted Majority Algorithm

Proof:

- Define potential function $\phi^{(t)} \triangleq \sum_{i=1}^{n} w_i^{(t)}$
- We will bound $\phi^{(T+1)}$ from below with any expert i's mistakes, and from above with WMA's mistakes
- For lower bound:

$$\phi^{(T+1)} = \sum_{j=1}^{n} w_j^{(T+1)} \geq w_i^{(T+1)} = \left(\frac{1}{2}\right)^{M_i^{(T)}}$$

- For upper bound, note that $\phi^{(1)} = n$ and any weight $w_i^{(1)} = 1$
  - Whenever WMA makes a mistake, we halve the weights for experts representing at least half of the total weights (since we follow the weighted majority)
  - This means that we lose at least $(\frac{1}{2} \cdot \frac{1}{2} =)\frac{1}{4}$ of the total weight from the previous t-th day

$$\phi^{(t+1)} \leq \frac{3}{4}\phi^{(t)}$$

Weighted Majority Algorithm

Proof (Cont.):

- This implies that we can bound the final value of the potential function by

$$\phi^{(T+1)} \leq \left(\frac{3}{4}\right)^{M_{\text{WMA}}^{(T)}} \cdot \phi^{(1)} = \left(\frac{3}{4}\right)^{M_{\text{WMA}}^{(T)}} \cdot n$$

- Combining both bounds together,

$$\left(\frac{1}{2}\right)^{M_i^{(T)}} \leq \phi^{(T+1)} \leq \left(\frac{3}{4}\right)^{M_{\text{WMA}}^{(T)}} \cdot n$$

- Taking the $\ln$ on both sides, we have

$$\begin{aligned} -M_i^{(T)} &\leq \log n + \log\left(\frac{3}{4}\right) \cdot M_{\text{WMA}}^{(T)} \\ M_{\text{WMA}}^{(T)} &\leq 2.41 \cdot (M_i^{(T)} + \log n) \end{aligned}$$

Multiplicative Weights Update Algorithm

- Aggregator makes a random decision instead of a deterministic decision
- Aggregator picks some distribution $\mathbf{p}^{(t)} = (p_1^{(t)}, ..., p_n^{(t)})$ over experts, where $p_i^{(t)}$ represents the probability of following expert i's prediction on the t-th day
- Adversary is still omniscient: with knowledge of the experts' prediction and of $\mathbf{p}^{(t)}$, it determines the costs $\mathbf{m}^{(t)} = (m_1^{(t)}, ..., m_n^{(t)}) \in [-1, 1]^n$, where $m_i^{(t)}$ is the cost of following expert i's prediction on the t-th day
- The expected cost on the t-th day is

$$\mathbb{E}[\text{Cost}^{(t)}] = \sum_{i=1}^{n} p_i^{(t)} \cdot m_i^{(t)} = \mathbf{p}^{(t)} \cdot \mathbf{m}^{(t)}$$

- Goal: Pick the distribution $\mathbf{p}^{(t)}$ on each t-th day to minimize the regret between the expected total cost and the minimum total cost of following the best expert

Multiplicative Weights Update Algorithm

Multiplicative Weights Update Algorithm (MWU)

- Assign a weight $w_i^{(1)} = 1$ to each expert i
- On each t-th day, pick the distribution $p_i^{(t)} = \frac{w_i^{(t)}}{\phi^{(t)}}$, where $\phi^{(t)} = \sum_{i=1}^{n} w_i^{(t)}$
- After observing $\mathbf{m}^{(t)}$, set $w_i^{(t+1)} \leftarrow w_i^{(t)} \cdot e^{-\epsilon m_i^{(t)}}$ for each expert i

- Note that $e^{-\epsilon m_i^{(t)}} < 1$, if $m_i^{(t)} > 0$. Otherwise, $e^{-\epsilon m_i^{(t)}} > 1$. Hence, the weights increase when it was profitable to follow the expert, and decrease when it was not
- Let $\text{Cost}_i = \sum_{t=1}^{T} m_i^{(t)}$ be the total cost of expert i, and Cost_{MWU} be the random total cost of MWU for all T days. Note that

$$\mathbb{E}[\text{Cost}_{\text{MWU}}] = \sum_{t=1}^{T} \mathbf{p}^{(t)} \cdot \mathbf{m}^{(t)}, \quad \mathbb{E}[\text{Regret}] = \mathbb{E}[\text{Cost}_{\text{MWU}}] - \min_{i=1,\dots,n} \text{Cost}_i$$

Multiplicative Weights Update Algorithm

Theorem

Suppose $\epsilon \leq 1$, and $\mathbf{p}^{(t)}$ is chosen by MWU for $t = 1, ..., T$. Then for any expert i:

$$\mathbb{E}[\text{Cost}_{\text{MWU}}] \leq \text{Cost}_i + \frac{\ln n}{\epsilon} + \epsilon T, \quad \mathbb{E}[\text{Regret}] \leq \frac{\ln n}{\epsilon} + \epsilon T$$

Proof:

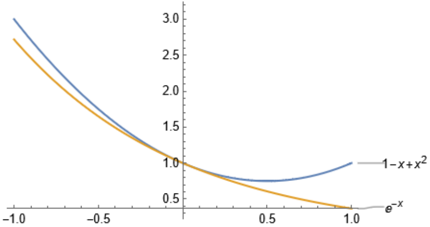
- For lower bound: $\phi^{(T+1)} = \sum_{i=1}^n w_j^{(T+1)} \geq w_i^{(T+1)} = w_i^{(1)} \cdot \prod_{t=1}^T e^{-\epsilon m_i^{(t)}} = e^{-\epsilon \sum_{i=1}^n m_i^{(t)}}$
- For upper bound:

$$\phi^{(t+1)} = \sum_{j=1}^{n} w_j^{(t+1)} = \sum_{j=1}^{n} w_j^{(t)} e^{-\epsilon m_j^{(t)}} \leq \sum_{j=1}^{n} w_j^{(t)} (1 - \epsilon m_j^{(t)} + \epsilon^2 (m_j^{(t)})^2)$$

Because $e^{-x} \leq 1 - x + x^2$ for $-1 \leq x \leq 1$

Multiplicative Weights Update Algorithm

Proof (Cont.):



$$
\begin{aligned}
\phi^{(t+1)} &\le \sum_{j=1}^{n} w_j^{(t)}(1-\epsilon m_j^{(t)} + \epsilon^2 (m_j^{(t)})^2) \le \sum_{j=1}^{n} w_j^{(t)}(1-\epsilon m_j^{(t)} + \epsilon^2) \\
&= \sum_{j=1}^{n} w_j^{(t)}(1+\epsilon^2) - \epsilon \sum_{j=1}^{n} w_j^{(t)} m_j^{(t)} = \phi^{(t)}(1+\epsilon^2) - \epsilon \sum_{j=1}^{n} \phi^{(t)} p_j^{(t)} m_j^{(t)} \\
&\le \phi^{(t)} \cdot e^{\epsilon^2 - \epsilon \sum_{j=1}^{n} p_j^{(t)} m_j^{(t)}} \qquad \text{(because } 1+x \le e^x\text{)}
\end{aligned}
$$

Multiplicative Weights Update Algorithm

Proof (Cont.):

- This implies that we can bound the final value of the potential function by

$$\phi^{(T+1)} \leq \phi^{(1)} \cdot e^{\epsilon^2 T - \epsilon \sum_{t=1}^{T} \sum_{j=1}^{n} p_j^{(t)} m_j^{(t)}} = n \cdot e^{\epsilon^2 T - \epsilon \sum_{t=1}^{T} \sum_{j=1}^{n} p_j^{(t)} m_j^{(t)}}$$

- Combining both bounds together,

$$e^{-\epsilon \sum_{t=1}^{T} m_j^{(t)}} \leq \phi^{(T+1)} \leq n \cdot e^{\epsilon^2 T - \epsilon \sum_{t=1}^{T} \mathbf{p}^{(t)} \cdot \mathbf{m}^{(t)}}$$

- Taking the $\ln$ on both sides, we have

$$-\epsilon \sum_{t=1}^{T} m_j^{(t)} \leq \ln n + \epsilon^2 T - \epsilon \sum_{t=1}^{T} \mathbf{p}^{(t)} \cdot \mathbf{m}^{(t)}$$

Improved Weighted Majority Algorithm

Theorem

After observing the outcome, for every incorrect expert i*, WMA' set* $w_i^{(t+1)} \leftarrow w_i^{(t)}(1-\epsilon)$
Suppose $\epsilon < \frac{1}{2}$*, then we have*

$$M_{\text{WMA}'}^{(T)} \leq 2(1+\epsilon) \cdot \sum_{t=1}^{T} M_i^{(T)} + \frac{2 \ln n}{\epsilon}$$

After observing the outcome, for every expert i*, MWU' set* $w_i^{(t+1)} \leftarrow w_i^{(t)}(1-\epsilon m_i^{(t)})$
Suppose $\epsilon < \frac{1}{2}$*, then we have*

$$\sum_{t=1}^{T} \mathbf{p}^{(t)} \cdot \mathbf{m}^{(t)} \leq \sum_{t=1}^{T} m_i^{(t)} + \epsilon \sum_{t=1}^{T} |m_i^{(t)}| + \frac{\ln n}{\epsilon}$$

Improved Weighted Majority Algorithm

Proof:

- Note that

$$\phi^{(t+1)} \leq \sum_{j \text{ is correct}} w_j^{(t)} + (1-\epsilon) \cdot \sum_{j \text{ is incorrect}} w_j^{(t)} = \sum_{j=1}^{n} w_j^{(t)} - \epsilon \cdot \sum_{j \text{ is incorrect}} w_j^{(t)} \leq (1-\frac{\epsilon}{2})\phi^{(t)}$$

- Hence, we have

$$(1-\epsilon)^{M_i^{(T)}} \leq \phi^{(T+1)} \leq \left(1-\frac{\epsilon}{2}\right)^{M_{\text{WMA}'}^{(T)}} \cdot n$$

- Note that $-x - x^2 < \ln(1-x) < -x$ for $0 < x < \frac{1}{2}$

$$\ln(1-\epsilon) \cdot M_i^{(T)} \leq \ln\left(1-\frac{\epsilon}{2}\right) \cdot M_{\text{WMA}'}^{(T)} + \ln n \;\Rightarrow\; (-\epsilon-\epsilon^2) \cdot M_i^{(T)} \leq \left(-\frac{\epsilon}{2}\right) \cdot M_{\text{WMA}'}^{(T)} + \ln n$$

Multiplicative Weights Update Algorithm

- Sometimes, it is useful to consider the average cost incurred per day
- Generalize the cost vector so that $\mathbf{m}^{(t)} = (m_1^{(t)}, ..., m_n^{(t)}) \in [-\rho, \rho]^n$
- The following theorem tells us that the average daily performance of MWU is as good as the best expert's average daily performance, within a linear term 2ϵ

Theorem

Suppose $\epsilon \leq 1$, *and* $\mathbf{p}^{(t)}$ *is chosen by MWU for* $t = 1, ..., T$
If $T \geq \frac{4\rho^2 \ln n}{\epsilon^2}$, *then for any expert* i

$$\frac{1}{T}\mathbb{E}[\text{Cost}_{\text{MWU}}] \leq \frac{1}{T}\text{Cost}_i + 2\epsilon$$

Application: Learning Linear Classifier

- Consider a set of k labeled examples $(\mathbf{a}_1, l_1), ..., (\mathbf{a}_k, l_k)$:
  - $\mathbf{a}_j = (a_{j,1}, ..., a_{j,n})$ is a n-dimensional feature vector and l_j is a label in $\{-1, 1\}$
- Goal: Find a linear classifier:
  - Unit vector $\mathbf{p} = (p_1, ..., p_n)$ such that $\sum_{j=1}^{n} p_j = 1$ and $l_j(\mathbf{a}_j \cdot \mathbf{p}) \geq 0$
- Define $\rho = \max_{j=1,...,k} \max_{i=1,...,n} |a_{j,n}|$

Learning Linear Classifier by MWU

- Initialize $w_i^{(1)} = 1$ for all i, and $\mathbf{p}^{(1)}$ accordingly
- At each t-th round, if there exists j such that $l_j(\mathbf{a}_j \cdot \mathbf{p}^{(t)}) < 0$ (i.e. $\mathbf{a}_j$ is not classified correctly), then
  - Set costs $\mathbf{m}^{(t)} = -\frac{l_j}{\rho}\mathbf{a}_j$, note that $\mathbf{m}^{(t)} \in [-1, 1]^n$
  - Run MWU to update $\mathbf{p}^{(t+1)}$ and proceed to the $(t+1)$-th round
- Otherwise, if there exists no j such that $l_j(\mathbf{a}_j \cdot \mathbf{p}^{(t)}) < 0$, then terminate

Application: Learning Linear Classifier

- Assume that there exists $\mathbf{p}^*$ such that $\sum_{j=1}^n p_j^* = 1$ and $l_j(\mathbf{a}_j \cdot \mathbf{p}^*) \geq \delta$ for some $\delta > 0$
- Note that for t (or some j)

$$\mathbf{m}^{(t)} \cdot \mathbf{p}^* = -\frac{l_j}{\rho}\mathbf{a}_j \cdot \mathbf{p}^* \leq -\frac{\delta}{\rho}$$

- Suppose the learning linear classifier algorithm terminates at the T-th round
- By MWU, we have

$$\begin{aligned}\sum_{t=1}^T \mathbf{p}^{(t)} \cdot \mathbf{m}^{(t)} &\leq \min_{i=1,\ldots,n} \sum_{t=1}^T m_i^{(t)} + \frac{\ln n}{\epsilon} + \epsilon T \\ &\leq \sum_{i=1}^n \sum_{t=1}^T m_i^{(t)} p_i^* + \frac{\ln n}{\epsilon} + \epsilon T \leq -\frac{\delta T}{\rho} + \frac{\ln n}{\epsilon} + \epsilon T\end{aligned}$$

Application: Learning Linear Classifier

- Note that when $t < T$ (before termination), we have $l_j(\mathbf{a}_j \cdot \mathbf{p}^{(t)}) < 0 \;\Rightarrow\; \mathbf{p}^{(t)} \cdot \mathbf{m}^{(t)} > 0$
- Hence, we have

$$0 < -\frac{\delta T}{\rho} + \frac{\ln n}{\epsilon} + \epsilon T$$

- If we set $\epsilon = \frac{\delta}{2\rho}$, then

$$T < \frac{4\rho^2 \ln n}{\delta^2}$$

- Namely, if there exists a linear classifier, then the learning linear classifier algorithm terminates, and that it finds it in less than $\frac{4\rho^2 \ln n}{\delta^2}$ rounds

Application: Boosting

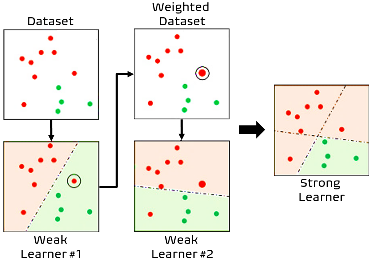
- Given a sequence of training data points $X = \{x_1, ..., x_n\}$ sampled from a universe set according to some (unknown) distribution $\mathcal{D}$
  - Each point has an (unknown) label $c(x_i) \in \{0, 1\}$
  - Find a hypothesis function $h \in \mathcal{C}$ that assigns labels to training data points, where the function h is taken from a set of functions (a concept class) $\mathcal{C}$ (e.g., the class of all linear classifiers), and predicts the function c in the best way possible (on average over $\mathcal{D}$)

- Strong learning algorithm: Output a hypothesis h, with probability at least $1-\delta$, such that

$$\mathbb{E}\Big[|h(x_i) - c(x_i)|\Big] \le \epsilon$$

- Weak learning algorithm: Output a hypothesis h, such that

$$\mathbb{E}\Big[|h(x_i) - c(x_i)|\Big] \le \frac{1}{2} - \gamma$$



Application: Boosting

- Goal: Use weak learning algorithms to construct a strong learning algorithm

AdaBoost

- For $t = 1, ..., T$ (where T is sufficiently large)
  - Use a weak learning algorithm to generate a hypothesis $h_t : X \to \{0, 1\}$
  - Compute the error of h_t: $E_t = \sum_{i=1}^{n} p_i^{(t)} |h_t(x_i) - c(x_i)|$
  - Set $\beta_t = \frac{E_t}{1-E_t}$
  - Set weight $w_i^{(t+1)} \leftarrow w_i^{(t)} \beta_t^{1-|h_t(x_i)-c(x_i)|}$ for each training data point x_i
  - Run MWU to update $\mathbf{p}^{(t+1)}$
- Output the final hypothesis $h : X \to \{0, 1\}$ based on weighted majority vote:

$$h(x) = \begin{cases} 1, & \text{if } \sum_{t=1}^{T} \log(\frac{1}{\beta_t}) h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \log(\frac{1}{\beta_t}) \\ 0, & \text{otherwise} \end{cases}$$

Regret vs. Competitive Ratio

- Online learning for regret minimization
  - Compare with the best expert (i.e. stationary offline optimal solution)
- Online algorithm for competitive ratio minimization
  - Compare with the best sequence of experts (i.e. dynamic offline optimal solution)
- Metrics
  - Regret: $\text{Cost}[\text{Algo}] - \text{Cost}[\text{Opt}]$
  - Competitive Ratio: $\frac{\text{Cost}[\text{Algo}]}{\text{Cost}[\text{Opt}]}$
  - Bounded regret $\Rightarrow$ bounded ratio

	Offline Optimal Comparsion	Metrics
Online Learning	Stationary (Weaker)	Difference (Stronger)👍
Online Algorithm	Dynamic (Stronger)👍	Ratio (Weaker)

References

Recommended Materials

- *The Multiplicative Weights Update Method: A Meta Algorithm and its Applications* (Arora, Hazan, Kale), Theory of Computing, 2012
- Watch online tutorial video about AdaBoost: `https://www.youtube.com/watch?v=LsK-xG1cLYA`