Multiplicative Weights Update

Si Yi (Cathy) Meng
June 26, 2019

UBC MLRG
Introduction
Introduction

Prediction from expert advise
Introduction

Prediction from expert advise

Figure 1: [Link to Groundhog Day 2018 image](https://www.cbc.ca/news/canada/windsor/groundhog-day-2018-1.4516220)
Introduction

Prediction from expert advise

Figure 1: https://www.cbc.ca/news/canada/windsor/groundhog-day-2018-1.4516220
Outline

- Theory
  - Weighted majority
  - Randomized weighted majority
  - General framework – Multiplicative weights update

- Applications
  - AdaBoost
  - Chernoff bounds
  - Online convex optimization
Theory
Problem Setup

- We are given the task of making a binary prediction on a sequence of events.
  - Rainy tomorrow? TSLA ↑ or ↓?
- We have access to $n$ experts.
- Each will predict 0 or 1 at a given time.
- Each expert has weight $w(t)$, representing its "credibility".
- At each timestep $t$, we somehow make a prediction based on the experts' predictions.
- The weights will be updated based on the correctness.
- Since we make no assumptions about the experts, we cannot guarantee an absolute level of quality of our predictions.
- Goal: do as well as the best expert in hindsight.
Problem Setup

- We are given the task of making a binary prediction on a sequence of events.
  - Rainy tomorrow? TSLA ↑ or ↓?
  - But we have no knowledge about it.
Problem Setup

- We are given the task of making a binary prediction on a sequence of events.
  - Rainy tomorrow? TSLA \( \uparrow \) or \( \downarrow \)?
  - But we have no knowledge about it.

- We have access to \( n \) experts
  - Each will predict 0 or 1 at a given time.
  - Each expert has weight \( w_i(t) \), representing its “credibility”.

Problem Setup

• We are given the task of making a binary prediction on a sequence of events.
  • Rainy tomorrow? TSLA ↑ or ↓?
  • But we have no knowledge about it.

• We have access to $n$ experts
  • Each will predict 0 or 1 at a given time.
  • Each expert has weight $w_i^{(t)}$, representing its “credibility”.

• At each timestep $t$, we somehow make a prediction based on the experts’ predictions.
Problem Setup

- We are given the task of making a binary prediction on a sequence of events.
  - Rainy tomorrow? TSLA ↑ or ↓?
  - But we have no knowledge about it.

- We have access to $n$ experts
  - Each will predict 0 or 1 at a given time.
  - Each expert has weight $w_{i(t)}$, representing its “credibility”.

- At each timestep $t$, we somehow make a prediction based on the experts’ predictions.

- The weights will be updated based on the correctness.
Problem Setup

- We are given the task of making a binary prediction on a sequence of events.
  - Rainy tomorrow? TSLA ↑ or ↓?
  - But we have no knowledge about it.
- We have access to $n$ experts
  - Each will predict 0 or 1 at a given time.
  - Each expert has weight $w_i^{(t)}$, representing its “credibility”.
- At each timestep $t$, we somehow make a prediction based on the experts’ predictions.
- The weights will be updated based on the correctness.
- Since we make no assumptions about the experts, we cannot guarantee an absolute level of quality of our predictions.
- **Goal:** do as well as the best expert in hindsight.
Weighted majority algorithm [2, 1]

- Set $w_i^{(1)} = 1$ for all $i$
- For $t = 1, 2, \ldots, T$
  - Experts make their decisions \{$x_1, \ldots, x_n$\}
  - We choose 1 if $\sum_{i:x_i=1} w_i^{(t)} \geq \sum_{i:x_i=0} w_i^{(t)}$ and 0 otherwise
  - Reveal the answer and incur a cost
  - Update weights
    - Incorrect experts: $w_i^{(t+1)} = (1 - \epsilon)w_i^{(t)}$
    - Correct experts: $w_i^{(t+1)} = w_i^{(t)}$
Theorem 1 ([2, 1])

After $T$ steps, let $m_i^{(T)}$ be the number of mistakes of expert $i$ and $M^{(T)}$ be the number of mistakes the weighted majority algorithm has made. Assuming $\epsilon \in (0, \frac{1}{2}]$, then we have the following bound:

$$M^{(T)} \leq \frac{2 \ln n}{\epsilon} + 2(1 + \epsilon)m_i^{(T)} \quad \forall i$$

In particular, this holds for $i = \text{the best expert}$, i.e. having the least $m_i^{(T)}$. 

Weighted majority
Proof.
It is clear that for all expert $i$, we have

$$w_i^{(T+1)} = (1 - \epsilon)m_i^{(T)}$$

(1)
Proof.
It is clear that for all expert $i$, we have

$$w_i^{(T+1)} = (1 - \epsilon)m_i^{(T)} \quad (1)$$

Let $\Phi^{(t)} = \sum_i w_i^{(t)}$ be the potential function, then $\Phi^{(1)} = n$. 
Weighted majority

Proof.
It is clear that for all expert $i$, we have

$$w_i^{(T+1)} = (1 - \epsilon)m_i^{(T)} \tag{1}$$

Let $\Phi(t) = \sum_i w_i^{(t)}$ be the potential function, then $\Phi(1) = n$. Each time we make a mistake, at least half of the weights decreases by $(1 - \epsilon)$. This implies $\Phi^{(t+1)} \leq \Phi^{(t)}\left(\frac{1}{2} + \frac{1}{2}(1 - \epsilon)\right) = \Phi^{(t)}(1 - \epsilon/2)$, which gives us

$$\Phi^{(T+1)} \leq n(1 - \epsilon/2)^{M^{(T)}} \tag{2}$$
Weighted majority

Since $\Phi(t) \geq w_i(t)$ for all $i$ and $t$, combining the above and applying

$$-\ln(1-x) \leq x + x^2 \quad \text{and} \quad \ln(x) \leq x - 1 \quad \text{for} \quad x \in (0, \frac{1}{2}]$$

gives us the desired bound

$$M(T) \leq \frac{2 \ln n}{\epsilon} + 2(1 + \epsilon) m_i(T) \quad \forall i.$$
Remarks:

- We made no assumptions on the sequence of events nor the quality of the experts.
Remarks:

- We made no assumptions on the sequence of events nor the quality of the experts.
- However, when $m_i^{(T)} \gg \frac{2 \ln n}{\epsilon}$, then from Theorem 1, the number of mistakes made by our algorithm will be upper bounded by approximately twice the number of mistakes made by the best expert.
Remarks:

- We made no assumptions on the sequence of events nor the quality of the experts.
- However, when $m_i^{(T)} \gg \frac{2\ln n}{\epsilon}$, then from Theorem 1, the number of mistakes made by our algorithm will be upper bounded by approximately twice the number of mistakes made by the best expert.
  - Tight for any deterministic algorithm.
  - Can remove the factor of 2 by a randomized version.
Randomized weighted majority

• Instead of deterministically following the majority, we randomly select an expert to follow with probability proportional to its weight.
Randomized weighted majority

- Instead of deterministically following the majority, we randomly select an expert to follow with probability proportional to its weight.
  - At the beginning, we select experts uniformly at random.
Randomized weighted majority

- Instead of deterministically following the majority, we randomly select an expert to follow with probability proportional to its weight.
  - At the beginning, we select experts uniformly at random.
  - As the events unfold, we lower the weights of the poorly performing ones, so they are less likely to be followed.
Randomized weighted majority

- Instead of deterministically following the majority, we randomly select an expert to follow with probability proportional to its weight.
  - At the beginning, we select experts uniformly at random.
  - As the events unfold, we lower the weights of the poorly performing ones, so they are less likely to be followed.
- If the events are chosen by an adversary, randomizing the selection of experts will improve our performance.
Randomized weighted majority

Randomized weighted majority algorithm[2, 1]

- Set $w_i^{(1)} = 1$ for all $i$
- For $t = 1, 2, \ldots, T$
  - Experts make their decisions $\{x_1, \ldots, x_n\}$
  - We choose $x_i$ with probability $p_i^{(t)} := \frac{w_i^{(t)}}{\sum_j w_j^{(t)}} = \frac{w_i^{(t)}}{\Phi(t)}$
  - Reveal the answer and incur a cost
  - Update weights
    - Incorrect experts: $w_i^{(t+1)} = (1 - \epsilon)w_i^{(t)}$
    - Correct experts: $w_i^{(t+1)} = w_i^{(t)}$
Theorem 2 ([2, 1])

After $T$ steps, let $m_i^{(T)}$ be the number of mistakes made by expert $i$. Assuming $\epsilon \in (0, \frac{1}{2}]$, then the expected number of mistakes $M^{(T)}$ made by the randomized weighted majority algorithm satisfies

$$M^{(T)} \leq \frac{\ln n}{\epsilon} + (1 + \epsilon)m_i^{(T)} \quad \forall i$$

Again, this holds for $i = \text{the best expert.}$
Proof.
Let $c_i(t) \in \{0, 1\}$ be the cost incurred by expert $i$ at time $t$. 
Proof.
Let $c_i^{(t)} \in \{0, 1\}$ be the cost incurred by expert $i$ at time $t$.
Then the expected cost of our algorithm at a particular timestep $t$ is
\[ \sum_i c_i^{(t)} p_i^{(t)} = \langle c^{(t)}, p^{(t)} \rangle. \]
After $T$ steps, we have
\[ M^{(T)} = \sum_{t=1}^{T} \langle c^{(t)}, p^{(t)} \rangle \] (3)
For the change in potential,

\[ \Phi^{(t+1)} = \sum_i w_i^{(t+1)} \]
For the change in potential,

\[
\Phi^{(t+1)} = \sum_i w_i^{(t+1)} \\
= \sum_i w_i^{(t)} (1 - \epsilon c_i^{(t)})
\]
Randomized weighted majority

For the change in potential,

\[ \Phi(t+1) = \sum_i w_i^{(t+1)} = \sum_i w_i^{(t)} (1 - \epsilon c_i^{(t)}) = \Phi(t) - \epsilon \sum_i \Phi(t) c_i^{(t)} p_i^{(t)} \]

By defn of \( p_i^{(t)} \)
Randomized weighted majority

For the change in potential,

\[ \Phi^{(t+1)} = \sum_i w_i^{(t+1)} \]

\[ = \sum_i w_i^{(t)} (1 - \epsilon c_i^{(t)}) \]

\[ = \Phi^{(t)} - \epsilon \sum_i \Phi^{(t)} c_i^{(t)} p_i^{(t)} \]  \[ \text{By defn of } p_i^{(t)} \]

\[ = \Phi^{(t)} (1 - \epsilon \langle c^{(t)}, p^{(t)} \rangle) \]
Randomized weighted majority

For the change in potential,

\[
\Phi^{(t+1)} = \sum_i w_i^{(t+1)} \\
= \sum_i w_i^{(t)} (1 - \epsilon c_i^{(t)}) \\
= \Phi^{(t)} - \epsilon \sum_i \Phi^{(t)} c_i^{(t)} p_i^{(t)} \\
= \Phi^{(t)} (1 - \epsilon \langle c^{(t)}, p^{(t)} \rangle) \\
\leq \Phi^{(t)} \exp(-\epsilon \langle c^{(t)}, p^{(t)} \rangle)
\]

where the last inequality comes from \(1 + x \leq \exp(x)\) for all \(x\).
Randomized weighted majority

By recursion, the potential after $T$ steps is then

$$\Phi^{(T+1)} \leq \Phi^{(1)} \prod_{t=1}^{T} \exp(-\epsilon \langle c^{(t)}, p^{(t)} \rangle)$$

(4)
Randomized weighted majority

By recursion, the potential after \( T \) steps is then

\[
\Phi^{(T+1)} \leq \Phi^{(1)} \prod_{t=1}^{T} \exp(-\epsilon \langle c^{(t)}, p^{(t)} \rangle) \\
= \Phi^{(1)} \exp(-\epsilon \sum_{t=1}^{T} \langle c^{(t)}, p^{(t)} \rangle)
\]

(4)
Randomized weighted majority

By recursion, the potential after $T$ steps is then

$$\Phi(T+1) \leq \Phi(1) \prod_{t=1}^{T} \exp(-\epsilon \langle c(t), p(t) \rangle)$$

$$= \Phi(1) \exp(-\epsilon \sum_{t=1}^{T} \langle c(t), p(t) \rangle)$$

$$= \Phi(1) \exp(-\epsilon M(T))$$

By equation 3

$$= n \exp(-\epsilon M(T))$$

(4)
Randomized weighted majority

By recursion, the potential after $T$ steps is then

$$\Phi(T+1) \leq \Phi(1) \prod_{t=1}^{T} \exp(-\epsilon \langle c(t), p(t) \rangle)$$

$$= \Phi(1) \exp(-\epsilon \sum_{t=1}^{T} \langle c(t), p(t) \rangle)$$

$$= \Phi(1) \exp(-\epsilon M^{(T)})$$

By equation 3

$$= n \exp(-\epsilon M^{(T)})$$

(4)

For each expert, its final weight is again given by

$$w_i^{(T+1)} = (1 - \epsilon)^{m_i^{(T)}} \leq \Phi(T+1)$$

(5)
Randomized weighted majority

By recursion, the potential after \( T \) steps is then

\[
\Phi^{(T+1)} \leq \Phi^{(1)} \prod_{t=1}^{T} \exp(-\epsilon \langle c^{(t)}, p^{(t)} \rangle )
\]

\[
= \Phi^{(1)} \exp(-\epsilon \sum_{t=1}^{T} \langle c^{(t)}, p^{(t)} \rangle )
\]

\[
= \Phi^{(1)} \exp(-\epsilon M^{(T)})
\]

By equation 3

\[
= n \exp(-\epsilon M^{(T)})
\]

(4)

For each expert, its final weight is again given by

\[
w_{i}^{(T+1)} = (1 - \epsilon)^{m_{i}^{(T)}} \leq \Phi^{(T+1)}
\]

(5)

Combining equations 4 and 5 and applying \( \ln(1 - \epsilon) \leq \epsilon(1 + \epsilon) \) gives us the desired bound

\[
M^{(T)} \leq \frac{\ln n}{\epsilon} + (1 + \epsilon)m_{i}^{(T)} \quad \forall i.
\]
Randomized weighted majority

\[ M^{(T)} \leq \frac{\ln n}{\epsilon} + (1 + \epsilon) m^{(T)}_i \quad \forall i. \]

- **Tradeoff**: by adjusting \( \epsilon \), we can make the “competitive ratio” of the algorithm as close to 1 as desired, at the expense of an increase in the additive constant.
Randomized weighted majority

\[ M^{(T)} \leq \frac{\ln n}{\epsilon} + (1 + \epsilon)m^{(T)} \quad \forall i. \]

- **Tradeoff**: by adjusting \( \epsilon \), we can make the “competitive ratio” of the algorithm as close to 1 as desired, at the expense of an increase in the additive constant.

- Denote \( m = m_i^{(T)} \). RHS is convex in \( \epsilon \) for \( \epsilon \in (0, \frac{1}{2}] \), take the derivative and set to 0, we get \( m = \epsilon^2 \ln n \)
  - Set \( \epsilon = \sqrt{\frac{\ln n}{m_i^{(T)}}} \)
  - Gives us the bound \( M^{(T)} \leq m + 2\sqrt{m \ln n} \)
Randomized weighted majority

\[ M^{(T)} \leq \frac{\ln n}{\epsilon} + (1 + \epsilon)m^{(T)}_i \quad \forall i. \]

- **Tradeoff**: by adjusting \( \epsilon \), we can make the “competitive ratio” of the algorithm as close to 1 as desired, at the expense of an increase in the additive constant.

- Denote \( m = m^{(T)}_i \). RHS is convex in \( \epsilon \) for \( \epsilon \in (0, \frac{1}{2}] \), take the derivative and set to 0, we get \( m = \epsilon^2 \ln n \)
  - Set \( \epsilon = \sqrt{(\ln n)/m^{(T)}_i} \)
  - Gives us the bound \( M^{(T)} \leq m + 2\sqrt{m \ln n} \)
  - But we don’t know \( m \)
Randomized weighted majority

\[ M^{(T)} \leq \frac{\ln n}{\epsilon} + (1 + \epsilon)m_i^{(T)} \quad \forall i. \]

- **Tradeoff**: by adjusting \( \epsilon \), we can make the “competitive ratio” of the algorithm as close to 1 as desired, at the expense of an increase in the additive constant.

- Denote \( m = m_i^{(T)} \). RHS is convex in \( \epsilon \) for \( \epsilon \in (0, \frac{1}{2}] \), take the derivative and set to 0, we get \( m = \epsilon^2 \ln n \)
  - Set \( \epsilon = \sqrt{(\ln n)/m_i^{(T)}} \)
  - Gives us the bound \( M^{(T)} \leq m + 2\sqrt{m \ln n} \)
  - But we don’t know \( m \)

- Guess and double trick: start with \( m = 4 \ln n \) and \( \epsilon = \frac{1}{2} \). Once every expert has made at least \( m \) mistakes, double \( m \) and update \( \epsilon = \frac{\sqrt{2}}{2} \).
Remarks:

- We now achieve a better bound with randomization.
Remarks:

- We now achieve a better bound with randomization.
- Only binary predictions/costs so far.
- Generalize to
  - A set of outcomes that are not necessarily binary.
  - Real-valued costs (within some range).
Multiplicative weights update algorithm [1]

- Set $w_i^{(1)} = 1$ for all $i$
- For $t = 1, 2, \ldots, T$
  - Experts make their decisions $\{x_1, \ldots, x_n\}$
  - We choose $x_i$ with probability $p_i^{(t)} := \frac{w_i^{(t)}}{\sum_j w_j^{(t)}} = \frac{w_i^{(t)}}{\Phi^{(t)}}$
  - Reveal the answer, incur costs $c_i^{(t)}$
  - Update weights for each expert $i$

$$w_i^{(t+1)} = w_i^{(t)}(1 - \epsilon c_i^{(t)})$$
**Theorem 3 ([1])**

Assume that all costs $c_i^{(t)} \in [-1, 1]$ and $\epsilon \in (0, \frac{1}{2}]$. After $T$ steps, let $m_i^{(T)}$ be the total cost of expert $i$, then the total expected cost $M^{(T)}$ made by the multiplicative weights algorithm satisfies

$$M^{(T)} \leq \sum_{t=1}^{T} c_i^{(t)} + \epsilon \sum_{t=1}^{T} |c_i^{(t)}| + \frac{\ln n}{\epsilon} \quad \forall i$$

Again, this holds for $i = \text{the best expert.}$
Proof.
Same as before, after $T$ steps, the total expected cost of our algorithm is

$$M^{(T)} = \sum_{t=1}^{T} \langle c^{(t)}, p^{(t)} \rangle.$$
Proof.
Same as before, after $T$ steps, the total expected cost of our algorithm is

$$M^{(T)} = \sum_{t=1}^{T} \langle c^{(t)}, p^{(t)} \rangle.$$ 

The change in potential is bounded the same way,

$$\Phi^{(t+1)} = \sum_{i} w_{i}^{(t+1)} \leq \Phi^{(t)} \exp(-\epsilon \langle c^{(t)}, p^{(t)} \rangle)$$
Proof.
Same as before, after $T$ steps, the total expected cost of our algorithm is

$$M^{(T)} = \sum_{t=1}^{T} \langle c(t), p(t) \rangle.$$ 

The change in potential is bounded the same way,

$$\Phi^{(t+1)} = \sum_{i} w_i^{(t+1)} \leq \Phi^{(t)} \exp(-\epsilon \langle c(t), p(t) \rangle)$$

which gives us

$$\Phi^{(T+1)} \leq \Phi^1 \exp(-\epsilon \sum_{t=1}^{T} \langle c(t), p(t) \rangle) = n \exp(-\epsilon M^{(T)})$$  \hspace{1cm} (6)
The following facts follow from the convexity of the exponential function:

\[(1 - \epsilon x) \geq (1 - \epsilon)^x \quad \text{if} \quad x \in [0, 1]\]

\[(1 - \epsilon x) \geq (1 + \epsilon)^{-x} \quad \text{if} \quad x \in [-1, 0]\]

By our assumption \(c_i^{(t)} \in [-1, 1]\), we have for every expert \(i\),

\[
 w_i^{(T+1)} = \prod_{t=1}^{T} (1 - \epsilon c_i^{(t)})
\]  

(7)
The following facts follow from the convexity of the exponential function:

\[
(1 - \epsilon x) \geq (1 - \epsilon)^{x} \quad \text{if } x \in [0, 1] \\
(1 - \epsilon x) \geq (1 + \epsilon)^{-x} \quad \text{if } x \in [-1, 0]
\]

By our assumption \(c_i^{(t)} \in [-1, 1]\), we have for every expert \(i\),

\[
\begin{align*}
\omega_i^{(T+1)} &= \prod_{t=1}^{T} (1 - \epsilon c_i^{(t)}) \\
&\geq (1 - \epsilon) \sum_{\geq 0} c_i^{(t)} \cdot (1 + \epsilon)^{- \sum_{< 0} c_i^{(t)}}
\end{align*}
\]

(7)

where the subscripts refer to \(t : c_i^{(t)} \geq 0\) and \(t : c_i^{(t)} < 0\), respectively.
Multiplicative weights update

Again, since $w_i^{(T+1)} \leq \Phi^{(T+1)}$, we can combine equation 6 and 7,

$$(1 - \epsilon)\sum_{i \geq 0} c_i^{(t)} \cdot (1 + \epsilon)\sum_{i < 0} c_i^{(t)} \leq n \exp(-\epsilon M^{(T)})$$
Again, since $w_i^{(T+1)} \leq \Phi^{(T+1)}$, we can combine equation 6 and 7,

$$(1 - \epsilon)\sum_{i \geq 0} c_i^{(t)} \cdot (1 + \epsilon)^{-\sum_{i < 0} c_i^{(t)}} \leq n \exp(-\epsilon M^{(T)})$$

Taking logs, negating, and rearranging,

$$\epsilon M^{(T)} \leq \ln n - \sum_{i \geq 0} c_i^{(t)} \ln(1 - \epsilon) + \sum_{i < 0} c_i^{(t)} \ln(1 + \epsilon)$$
Multiplicative weights update

Again, since \( w_i^{(T+1)} \leq \Phi^{(T+1)} \), we can combine equation 6 and 7,

\[
(1 - \epsilon) \sum_{i \geq 0} c_i^{(t)} \cdot (1 + \epsilon)^{-\sum_{i < 0} c_i^{(t)}} \leq n \exp(-\epsilon M^{(T)})
\]

Taking logs, negating, and rearranging,

\[
\epsilon M^{(T)} \leq \ln n - \sum_{i \geq 0} c_i^{(t)} \ln(1 - \epsilon) + \sum_{i < 0} c_i^{(t)} \ln(1 + \epsilon)
\]

Apply \(-\ln(1 - x) \leq x + x^2\) and \(\ln(1 + x) \geq x + x^2\) for \(x \leq \frac{1}{2}\),

\[
\epsilon M^{(T)} \leq \ln n + \sum_{i \geq 0} c_i^{(t)}(\epsilon + \epsilon^2) + \sum_{i < 0} c_i^{(t)}(\epsilon - \epsilon^2)
\]
Multiplicative weights update

Again, since \( w_i^{(T+1)} \leq \Phi^{(T+1)} \), we can combine equation 6 and 7,

\[
(1 - \epsilon)\sum_{i \geq 0} c_i^{(t)} \cdot (1 + \epsilon)^{- \sum_{i < 0} c_i^{(t)}} \leq n \exp(-\epsilon M^{(T)})
\]

Taking logs, negating, and rearranging,

\[
\epsilon M^{(T)} \leq \ln n - \sum_{i \geq 0} c_i^{(t)} \ln(1 - \epsilon) + \sum_{i < 0} c_i^{(t)} \ln(1 + \epsilon)
\]

Apply \(-\ln(1 - x) \leq x + x^2\) and \(\ln(1 + x) \geq x + x^2\) for \(x \leq \frac{1}{2}\),

\[
\epsilon M^{(T)} \leq \ln n + \sum_{i \geq 0} c_i^{(t)} (\epsilon + \epsilon^2) + \sum_{i < 0} c_i^{(t)} (\epsilon - \epsilon^2)
\]

\[
\epsilon M^{(T)} \leq \ln n + \epsilon \sum_{t=1}^{T} c_i^{(t)} + \epsilon^2 \sum_{t=1}^{T} |c_i^{(t)}|
\]

Dividing \(\epsilon\) on both sides gives us the desired bound. \(\square\)
Comparison

\[ \text{WM} : M^{(t)} \leq 2(1 + \epsilon)m_i^{(T)} + \frac{2 \ln n}{\epsilon} \]

\[ \text{RWM} : M^{(t)} \leq (1 + \epsilon)m_i^{(T)} + \frac{\ln n}{\epsilon} \]

\[ \text{MWU} : M^{(t)} \leq \sum_{t=1}^{T} c_i^{(t)} + \epsilon \sum_{t=1}^{T} |c_i^{(t)}| + \frac{\ln n}{\epsilon} \]

Can further generalize to the Matrix Multiplicative Weights algorithm:

- Cost vectors → cost matrices
- Probability vectors → density matrices
- Mainly applied in solving SDPs.
Comparison

\[
WM : M^{(t)} \leq 2(1 + \epsilon)m_i^{(T)} + \frac{2 \ln n}{\epsilon}
\]

\[
RWM : M^{(t)} \leq (1 + \epsilon)m_i^{(T)} + \frac{\ln n}{\epsilon}
\]

\[
MWU : M^{(t)} \leq \sum_{t=1}^{T} c_i^{(t)} + \epsilon \sum_{t=1}^{T} |c_i^{(t)}| + \frac{\ln n}{\epsilon}
\]

Can further generalize to the Matrix Multiplicative Weights algorithm:

- Cost vectors \(\rightarrow\) cost matrices
- Probability vectors \(\rightarrow\) density matrices
- Mainly applied in solving SDPs.
Application
AdaBoost

- **Adaptive Boosting**, Freund and Schapire 1996 [3].
- Classification problems: $x_i \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$, $i = 1, \ldots, n$
- **Goal**: combine a set of $T$ weak classifiers into a strong one.
AdaBoost

Algorithm AdaBoost

Input: sequence of $N$ labeled examples $\langle (x_1, y_1), \ldots, (x_N, y_N) \rangle$
- distribution $D$ over the $N$ examples
- weak learning algorithm WeakLearn
- integer $T$ specifying number of iterations

Initialize the weight vector: $w_i^1 = D(i)$ for $i = 1, \ldots, N$.

Do for $t = 1, 2, \ldots, T$

1. Set

$$p^t = \frac{w^t}{\sum_{i=1}^{N} w_i^t}$$

2. Call WeakLearn, providing it with the distribution $p^t$; get back a hypothesis $h_t: X \rightarrow [0, 1]$.

3. Calculate the error of $h_t$: $\epsilon_t = \sum_{i=1}^{N} p_i^t |h_t(x_i) - y_i|$.

4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$.

5. Set the new weights vector to be

$$w_i^{t+1} = w_i^t \beta_i^{1-|h_t(x_i) - y_i|}$$

Output the hypothesis

$$h_f(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^{T} \left( \log \frac{1}{\beta_i} \right) h_t(x) \geq \frac{1}{2} \sum_{i=1}^{T} \log \frac{1}{\beta_i} \\ 0 & \text{otherwise.} \end{cases}$$

Figure 2: AdaBoost algorithm [3]
AdaBoost

Figure 3: https://bit.ly/31UrxIo
Chernoff Bounds

- Let $X = \sum_{i=1}^{T} X_i$ be the sum of $n$ independent random variables $X_i \in (0, 1]$, and $\mu = \mathbb{E}X$.

- Chernoff bounds show that $X$ is sharply concentrated about $\mu$:

$$\Pr(X \geq (1 + \delta)\mu) \leq \left(\frac{\exp(\delta)}{(1 + \delta)^{1+\delta}}\right)^\mu \text{ and } \Pr(X \leq (1 - \delta)\mu) \leq \left(\frac{\exp(-\delta)}{(1 - \delta)^{1-\delta}}\right)^\mu$$
Chernoff Bounds

- Let \( X = \sum_{i=1}^{T} X_i \) be the sum of \( n \) independent random variables \( X_i \in (0, 1] \), and \( \mu = \mathbb{E}X \).

- **Chernoff bounds** show that \( X \) is sharply concentrated about \( \mu \):

\[
P(X \geq (1 + \delta)\mu) \leq \left( \frac{\exp(\delta)}{(1 + \delta)^{1+\delta}} \right)^\mu \quad \text{and} \quad P(X \leq (1 - \delta)\mu) \leq \left( \frac{\exp(-\delta)}{(1 - \delta)^{1-\delta}} \right)^\mu
\]

- By Markov’s inequality,

\[
P(X \geq a) = P(\exp(tX) \geq \exp(ta)) \leq \frac{\mathbb{E}[\exp(tX)]}{\exp(ta)} = \frac{\mathbb{E}[\exp(t \sum_i X_i)]}{\exp(ta)}
\]
Chernoff Bounds

• Let \( X = \sum_{i=1}^{T} X_i \) be the sum of \( n \) independent random variables \( X_i \in (0, 1] \), and \( \mu = \mathbb{E}X \).

• Chernoff bounds show that \( X \) is sharply concentrated about \( \mu \):

\[
\mathbb{P}(X \geq (1 + \delta)\mu) \leq \left( \frac{\exp(\delta)}{(1 + \delta)^{1+\delta}} \right)^{\mu} \quad \text{and} \quad \mathbb{P}(X \leq (1 - \delta)\mu) \leq \left( \frac{\exp(-\delta)}{(1 - \delta)^{1-\delta}} \right)^{\mu}
\]

• By Markov’s inequality,

\[
P(X \geq a) = P(\exp(tX) \geq \exp(ta)) \leq \frac{\mathbb{E}[\exp(tX)]}{\exp(ta)} = \frac{\mathbb{E}[\exp(t \sum_i X_i)]}{\exp(ta)}
\]

• Young pointed out in 1995[4]: at every step \( i \), we receive \( X_i \) and multiplicatively update the “potential” by \( \exp(tX_i) \)
• Decision set is a convex, compact set $\mathcal{K} \subseteq \mathbb{R}^n$
Online convex optimization

- Decision set is a convex, compact set $\mathcal{K} \subseteq \mathbb{R}^n$
- We need to minimize a convex function $f(t)$ at each time $t$ by choosing a point $p(t) \in \mathcal{K}$, and incur cost $f(t)(p(t))$
Online convex optimization

- Decision set is a convex, compact set $\mathcal{K} \subseteq \mathbb{R}^n$
- We need to minimize a convex function $f^{(t)}$ at each time $t$ by choosing a point $p^{(t)} \in \mathcal{K}$, and incur cost $f^{(t)}(p^{(t)})$
- Goal is to minimize regret:
  \[
  R^{(T)} = \sum_{t=1}^{T} f^{(t)}(p^{(t)}) - \min_{p \in \mathcal{K}} \sum_{t=1}^{T} f^{(t)}(p)
  \]
Online convex optimization

- Decision set is a convex, compact set $\mathcal{K} \subseteq \mathbb{R}^n$
- We need to minimize a convex function $f^{(t)}$ at each time $t$ by choosing a point $p^{(t)} \in \mathcal{K}$, and incur cost $f^{(t)}(p^{(t)})$
- Goal is to minimize regret:
  \[ R^{(T)} = \sum_{t=1}^{T} f^{(t)}(p^{(t)}) - \min_{p \in \mathcal{K}} \sum_{t=1}^{T} f^{(t)}(p) \]
- To use the MWU method for the special case where $\mathcal{K}$ is the $n$-dimensional simplex,
  - Define $\rho = \max_{p \in \mathcal{K}} \max_{t} \| \nabla f^{(t)}(p) \|_\infty$
  - Then run MWU with $\epsilon = \sqrt{\ln n / T}$ and costs $c^{(t)} := \frac{1}{\rho} \nabla f^{(t)}(p^{(t)})$, where $\rho$ is to make sure $c^{(t)}_i \in [-1, 1]$
  - Can show that $R^{(T)} \leq 2\rho \sqrt{T \ln n}$ after $T$ rounds.
Conclusion

Summary:

- General framework of multiplicative weights update method.
- Prediction from expert advice with performance competitive to the best expert in hindsight.
- Relationship to other areas.

The multiplicative weights update method: a meta-algorithm and applications.


A. Blum.

On-line algorithms in machine learning.


Y. Freund and R. E. Schapire.

A decision-theoretic generalization of on-line learning and an application to boosting.


N. E. Young.

Randomized rounding without solving the linear program.

1995.
Thank you

Only in America...

do we accept weather predictions from a rodent but deny climate change evidence from scientists.