YOLO ALGORITHM

Lecture 13: Multiplicative Weights Update

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

October 28, 2020

Overview

- Multiplicative Weights Update
- Solving Linear Programs
- Conclusion
- Acknowledgements

Learning from Experts

Setup: investing your co-op money on stock markets (or gambling in sports matches).

Learning from Experts

Setup: investing your co-op money on stock markets (or gambling in sports matches).

- Objective: to get rich, but we don't know much about stock markets (or sports matches)

Learning from Experts

Setup: investing your co-op money on stock markets (or gambling in sports matches).

- Objective: to get rich, but we don't know much about stock markets (or sports matches)
- We have access to n experts (say from news programs, newspapers, social media, etc)

Learning from Experts

Setup: investing your co-op money on stock markets (or gambling in sports matches).

- Objective: to get rich, but we don't know much about stock markets (or sports matches)
- We have access to n experts (say from news programs, newspapers, social media, etc)
  - Each morning, before market opens, experts predict whether the price of a stock will go up or down

Learning from Experts

Setup: investing your co-op money on stock markets (or gambling in sports matches).

- Objective: to get rich, but we don't know much about stock markets (or sports matches)
- We have access to n experts (say from news programs, newspapers, social media, etc)
  - Each morning, before market opens, experts predict whether the price of a stock will go up or down
  - By the time market closes, we can check the outcome, who was right or wrong.

Learning from Experts

Setup: investing your co-op money on stock markets (or gambling in sports matches).

- Objective: to get rich, but we don't know much about stock markets (or sports matches)
- We have access to n experts (say from news programs, newspapers, social media, etc)
  - Each morning, before market opens, experts predict whether the price of a stock will go up or down
  - By the time market closes, we can check the outcome, who was right or wrong.
    - Experts who were right earn one dollar
    - Experts who were wrong lose one dollar

Learning from Experts

Setup: investing your co-op money on stock markets (or gambling in sports matches).

- Objective: to get rich, but we don't know much about stock markets (or sports matches)
- We have access to n experts (say from news programs, newspapers, social media, etc)
  - Each morning, before market opens, experts predict whether the price of a stock will go up or down
  - By the time market closes, we can check the outcome, who was right or wrong.
    - Experts who were right earn one dollar
    - Experts who were wrong lose one dollar
- Some expert did really well, and if we followed their advice we would have made a lot of money...

Learning from Experts

Setup: investing your co-op money on stock markets (or gambling in sports matches).

- Objective: to get rich, but we don't know much about stock markets (or sports matches)
- We have access to n experts (say from news programs, newspapers, social media, etc)
  - Each morning, before market opens, experts predict whether the price of a stock will go up or down
  - By the time market closes, we can check the outcome, who was right or wrong.
    - Experts who were right earn one dollar
    - Experts who were wrong lose one dollar
- Some expert did really well, and if we followed their advice we would have made a lot of money...
- Hindsight is 20/20 though. To make money, need to make a decision on what & how to trade every day.

Online algorithms!

Learning from Experts

Setup: investing your co-op money on stock markets (or gambling in sports matches).

- Objective: to get rich, but we don't know much about stock markets (or sports matches)
- We have access to n experts (say from news programs, newspapers, social media, etc)
  - Each morning, before market opens, experts predict whether the price of a stock will go up or down
  - By the time market closes, we can check the outcome, who was right or wrong.
    - Experts who were right earn one dollar
    - Experts who were wrong lose one dollar
- Some expert did really well, and if we followed their advice we would have made a lot of money...
- Hindsight is 20/20 though. To make money, need to make a decision on what & how to trade every day.
- Can we hope to do *as well as the best expert* in hindsight?

Other Applications

- Online Learning

Other Applications

- Online Learning
  1. Experts are weak classifiers, want to choose hypothesis based on these experts

Other Applications

- Online Learning
  1. Experts are weak classifiers, want to choose hypothesis based on these experts
  2. Boosting (in learning theory)

Other Applications

- Online Learning
  1. Experts are weak classifiers, want to choose hypothesis based on these experts
  2. Boosting (in learning theory)
- Solving linear programs! (today)

Other Applications

- Online Learning
  1. Experts are weak classifiers, want to choose hypothesis based on these experts
  2. Boosting (in learning theory)
- Solving linear programs! (today)
- Game Theory (where this algorithm first appeared - even though discovered independently)

Other Applications

- Online Learning
  1. Experts are weak classifiers, want to choose hypothesis based on these experts
  2. Boosting (in learning theory)
- Solving linear programs! (today)
- Game Theory
- many more

Why this Benchmark?

Say we are trading for T days.

Why not want our algorithm to make as much money as a function of T as we can?

Why this Benchmark?

Say we are trading for T days.

Why not want our algorithm to make as much money as a function of T as we can?

- With the benchmark above, guessing correctly (in expectation) $T/2$ times is *trivial* (pick your trades at random)

Why this Benchmark?

Say we are trading for T days.

Why not want our algorithm to make as much money as a function of T as we can?

- With the benchmark above, guessing correctly (in expectation) $T/2$ times is *trivial* (pick your trades at random)
- It turns out, $T/2$ correct guesses (in expectation) is also *optimal*

in worst case

Why this Benchmark?

Say we are trading for T days.

Why not want our algorithm to make as much money as a function of T as we can?

- With the benchmark above, guessing correctly (in expectation) $T/2$ times is *trivial* (pick your trades at random)
- It turns out, $T/2$ correct guesses (in expectation) is also *optimal*
    - Worst-case analysis.

Warm-up Idea

Say we knew that there was *one expert* which will be *right every time*.
What should we do?

Warm-up Idea

Say we knew that there was *one expert* which will be *right every time*.
What should we do?

- At each trading day, take *majority vote* of the opinions of the experts.

Warm-up Idea

Say we knew that there was *one expert* which will be *right every time*. What should we do?

- At each trading day, take *majority vote* of the opinions of the experts.
- If we made the right trade, do nothing.

Warm-up Idea

Say we knew that there was *one expert* which will be *right every time*.
What should we do?

- At each trading day, take *majority vote* of the opinions of the experts.
- If we made the right trade, do nothing.
- If our trade was bad, at the end of the trading day *discard* all experts that *made a mistake that day*.

Warm-up Idea

Say we knew that there was *one expert* which will be *right every time*. What should we do?

- At each trading day, take *majority vote* of the opinions of the experts.
- If we made the right trade, do nothing.
- If our trade was bad, at the end of the trading day *discard* all experts that *made a mistake that day*.

1. Every time we made a bad trade, we discard half of the experts.
    - We took majority vote, so at least half the experts also made bad trades

Warm-up Idea

Say we knew that there was *one expert* which will be *right every time*. What should we do?

- At each trading day, take *majority vote* of the opinions of the experts.
- If we made the right trade, do nothing.
- If our trade was bad, at the end of the trading day *discard* all experts that *made a mistake that day*.

1. Every time we made a bad trade, we discard half of the experts.
    - We took majority vote, so at least half the experts also made bad trades
2. After $\log n$ bad trades, only the expert who is always right will remain! From then on, we will always be right!

Warm-up Idea

Say we knew that there was *one expert* which will be *right every time*. What should we do?

- At each trading day, take *majority vote* of the opinions of the experts.
- If we made the right trade, do nothing.
- If our trade was bad, at the end of the trading day *discard* all experts that *made a mistake that day*.

1. Every time we made a bad trade, we discard half of the experts.
    - We took majority vote, so at least half the experts also made bad trades
2. After $\log n$ bad trades, only the expert who is always right will remain! From then on, we will always be right!

Total money we made: $\geq T - \log n$

Total money best expert made: T

Multiplicative Weights Update Algorithm

- In general do not have a single expert who is right all the time.

Multiplicative Weights Update Algorithm

- In general do not have a single expert who is right all the time.
- Algorithm from previous slide *not robust* to having “almost perfect” experts (say experts that make one mistake)

Multiplicative Weights Update Algorithm

- In general do not have a single expert who is right all the time.
- Algorithm from previous slide *not robust* to having “almost perfect” experts (say experts that make one mistake)
- There is a way of making previous idea robust:

  Whenever an expert makes a mistake, “consider their opinions with less importance.”

Multiplicative Weights Update Algorithm

- In general do not have a single expert who is right all the time.
- Algorithm from previous slide *not robust* to having "almost perfect" experts (say experts that make one mistake)
- There is a way of making previous idea robust:

  Whenever an expert makes a mistake, "consider their opinions with less importance."

1. Let $w_t : [n] \to \mathbb{R}_+$ be a function from each expert to the non-negative reals, and $0 < \varepsilon < 1$
   - $w_t(i)$ is the *weight* of expert i at time t

Multiplicative Weights Update Algorithm

- In general do not have a single expert who is right all the time.
- Algorithm from previous slide *not robust* to having "almost perfect" experts (say experts that make one mistake)
- There is a way of making previous idea robust:

  Whenever an expert makes a mistake, "consider their opinions with less importance."

1. Let $w_t : [n] \to \mathbb{R}_+$ be a function from each expert to the non-negative reals, and $0 < \varepsilon < 1$
   - $w_t(i)$ is the *weight* of expert i at time t
2. In beginning every expert has weight $w_1(i) = 1$

Multiplicative Weights Update Algorithm

- In general do not have a single expert who is right all the time.
- Algorithm from previous slide *not robust* to having "almost perfect" experts (say experts that make one mistake)
- There is a way of making previous idea robust:

  Whenever an expert makes a mistake, "consider their opinions with less importance."

1. Let $w_t : [n] \to \mathbb{R}_+$ be a function from each expert to the non-negative reals, and $0 < \varepsilon < 1$
   - $w_t(i)$ is the *weight* of expert i at time t
2. In beginning every expert has weight $w_1(i) = 1$
3. If an expert makes a mistake at day t, make $w_{t+1}(i) = \underbrace{w_t(i) \cdot (1 - \varepsilon)}_{\text{decreasing weight of } i}$

Multiplicative Weights Update Algorithm

- In general do not have a single expert who is right all the time.
- Algorithm from previous slide *not robust* to having "almost perfect" experts (say experts that make one mistake)
- There is a way of making previous idea robust:

  Whenever an expert makes a mistake, "consider their opinions with less importance."

1. Let $w_t : [n] \to \mathbb{R}_+$ be a function from each expert to the non-negative reals, and $0 < \varepsilon < 1$
   - $w_t(i)$ is the *weight* of expert i at time t
2. In beginning every expert has weight $w_1(i) = 1$
3. If an expert makes a mistake at day t, make $w_{t+1}t(i) = w_t(i) \cdot (1 - \varepsilon)$
4. Each trading day, choose to trade based on *weighted majority* of the decisions of the experts

Multiplicative Weights Update Algorithm

Algorithm:

Multiplicative Weights Update Algorithm

Algorithm:

1. **Setup:** we have a binary decision to make (i.e., $\{-1, +1\}$) and we have access to n experts, indexed by the set $[n]$.

   $= \{1, 2, \ldots, n\}$

Multiplicative Weights Update Algorithm

Algorithm:

1. **Setup:** we have a binary decision to make (i.e., $\{-1, +1\}$) and we have access to n experts, indexed by the set $[n]$.
   - At each time step t expert i takes decision $d_t(i) \in \{-1, +1\}$
   - Parameter $0 < \varepsilon < 1/2$

Multiplicative Weights Update Algorithm

Algorithm:

1. **Setup:** we have a binary decision to make (i.e., $\{-1, +1\}$) and we have access to n experts, indexed by the set $[n]$.
   - At each time step i, expert takes decision $d_t(i) \in \{-1, +1\}$
   - Parameter $0 < \varepsilon < 1/2$
2. Let $w_t : [n] \rightarrow \mathbb{R}_+$ weight function
   - $w_t(i)$ is the *weight* of expert i at time t
   - In beginning every expert has weight $w_1(i) = 1$

Multiplicative Weights Update Algorithm

Algorithm:

1. **Setup:** we have a binary decision to make (i.e., $\{-1, +1\}$) and we have access to n experts, indexed by the set $[n]$.
   - At each time step i, expert takes decision $d_t(i) \in \{-1, +1\}$
   - Parameter $0 < \varepsilon < 1/2$
2. Let $w_t : [n] \to \mathbb{R}_+$ weight function
   - $w_t(i)$ is the *weight* of expert i at time t
   - In beginning every expert has weight $w_1(i) = 1$
3. At each time step (i.e. for $t = 1, \ldots, T$):

Multiplicative Weights Update Algorithm

Algorithm:

1. **Setup:** we have a binary decision to make (i.e., $\{-1, +1\}$) and we have access to n experts, indexed by the set $[n]$.
   - At each time step i, expert takes decision $d_t(i) \in \{-1, +1\}$
   - Parameter $0 < \varepsilon < 1/2$
2. Let $w_t : [n] \to \mathbb{R}_+$ weight function
   - $w_t(i)$ is the *weight* of expert i at time t
   - In beginning every expert has weight $w_1(i) = 1$
3. At each time step (i.e. for $t = 1, \ldots, T$):
   1. Make your decision based on *weighted majority*:

$$\begin{cases} +1, & \text{if } \underbrace{\sum_{i=1}^{n} w_t(i) \cdot d_t(i)}_{\Delta} \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

$$\Delta = \overbrace{\sum_{d_t(i)=+1} w_t(i)}\; - \overbrace{\sum_{d_t(i)=-1} w_t(i)}$$

Multiplicative Weights Update Algorithm

Algorithm:

1. **Setup:** we have a binary decision to make (i.e., $\{-1, +1\}$) and we have access to n experts, indexed by the set $[n]$.
   - At each time step i, expert takes decision $d_t(i) \in \{-1, +1\}$
   - Parameter $0 < \varepsilon < 1/2$
2. Let $w_t : [n] \to \mathbb{R}_+$ weight function
   - $w_t(i)$ is the *weight* of expert i at time t
   - In beginning every expert has weight $w_1(i) = 1$
3. At each time step (i.e. for $t = 1, \ldots, T$):
   1. Make your decision based on *weighted majority*:
   $$\begin{cases} +1, & \text{if } \sum_{i=1}^{n} w_t(i) \cdot d_t(i) \geq 0 \\ -1, & \text{otherwise} \end{cases}$$
   2. If an expert makes a mistake at time t, make
   $$w_{t+1}(i) = \boxed{w_t(i) \cdot (1 - \varepsilon)}$$

Analysis

Theorem (Multiplicative Weights Update)

Let M_t be the number of mistakes that our algorithm makes until time t, and let $M_t(i)$ be the number of mistakes that expert i made until time t. Then, for any expert $i \in [n]$, we have:

$$M_t \leq 2 \cdot (1+\varepsilon) M_t(i) + \frac{2 \log n}{\varepsilon}$$

M_t = # mistakes we make up to t

$M_t(i)$ = " " expert i makes

we make mistakes comparably to best expert

Analysis

Theorem (Multiplicative Weights Update)

Let M_t be the number of mistakes that our algorithm makes until time t, and let $M_t(i)$ be the number of mistakes that expert i made until time t. Then, for any expert $i \in [n]$, we have:

$$M_t \leq 2 \cdot (1+\varepsilon) M_t(i) + \frac{2 \log n}{\varepsilon}$$

- Potential function

$$\Phi_t = \sum_{i=1}^{n} w_t(i)$$

Analysis

Theorem (Multiplicative Weights Update)

Let M_t be the number of mistakes that our algorithm makes until time t, and let $M_t(i)$ be the number of mistakes that expert i made until time t. Then, for any expert $i \in [n]$, we have:

$$M_t \leq 2 \cdot (1+\varepsilon) M_t(i) + \frac{2 \log n}{\varepsilon}$$

- Potential function

$$\Phi_t = \sum_{i=1}^{n} w_t(i)$$

- **Intuition:**

Analysis

Theorem (Multiplicative Weights Update)

Let M_t be the number of mistakes that our algorithm makes until time t, and let $M_t(i)$ be the number of mistakes that expert i made until time t. Then, for any expert $i \in [n]$, we have:

$$M_t \leq 2 \cdot (1+\varepsilon) M_t(i) + \frac{2 \log n}{\varepsilon}$$

- Potential function

$$\Phi_t = \sum_{i=1}^{n} w_t(i)$$

- **Intuition:**
    - If we make mistake, Φ_{t+1} decreases by a multiplicative factor w.r.t. Φ_t

Analysis

Theorem (Multiplicative Weights Update)

Let M_t be the number of mistakes that our algorithm makes until time t, and let $M_t(i)$ be the number of mistakes that expert i made until time t. Then, for any expert $i \in [n]$, we have:

$$M_t \leq 2 \cdot (1+\varepsilon) M_t(i) + \frac{2 \log n}{\varepsilon}$$

- Potential function

$$\Phi_t = \sum_{i=1}^{n} w_t(i)$$

- **Intuition:**
  - If we make mistake, Φ_{t+1} decreases by a multiplicative factor w.r.t. Φ_t
  - Φ_t is monotonically decreasing (so if we get it right, potential does not increase either)

Analysis

Theorem (Multiplicative Weights Update)

Let M_t be the number of mistakes that our algorithm makes until time t, and let $M_t(i)$ be the number of mistakes that expert i made until time t. Then, for any expert $i \in [n]$, we have:

$$M_t \leq 2 \cdot (1 + \varepsilon) M_t(i) + \frac{2 \log n}{\varepsilon}$$

- Potential function

$$\Phi_t = \sum_{i=1}^{n} w_t(i)$$

- **Intuition:**
  - If we make mistake, Φ_{t+1} decreases by a multiplicative factor w.r.t. Φ_t
  - Φ_t is monotonically decreasing (so if we get it right, potential does not increase either)
  - Initially $\Phi_1 = n$
  - $\Phi_t \geq 0$ for all t

Analysis

Theorem (Multiplicative Weights Update)

Let M_t be the number of mistakes that our algorithm makes until time t, and let $M_t(i)$ be the number of mistakes that expert i made until time t. Then, for any expert $i \in [n]$, we have:

$$M_t \leq 2 \cdot (1 + \varepsilon) M_t(i) + \frac{2 \log n}{\varepsilon}$$

Analysis

Theorem (Multiplicative Weights Update)

Let M_t be the number of mistakes that our algorithm makes until time t, and let $M_t(i)$ be the number of mistakes that expert i made until time t. Then, for any expert $i \in [n]$, we have:

$$M_t \leq 2 \cdot (1+\varepsilon) M_t(i) + \frac{2 \log n}{\varepsilon}$$

- Potential function $\Phi_t = \sum_{i=1}^{n} w_t(i)$

Analysis

Theorem (Multiplicative Weights Update)

Let M_t be the number of mistakes that our algorithm makes until time t, and let $M_t(i)$ be the number of mistakes that expert i made until time t. Then, for any expert $i \in [n]$, we have:

$$M_t \leq 2 \cdot (1+\varepsilon) M_t(i) + \frac{2 \log n}{\varepsilon}$$

- Potential function $\Phi_t = \sum_{i=1}^{n} w_t(i)$
- If we made a mistake, at least half the weight was on the wrong answer. Thus

don't change   wrong

$$\Phi_{t+1} = \boxed{\sum_{i \text{ right}} w_t(i)} + (1-\varepsilon) \cdot \sum_{j \text{ wrong}} w_t(j) \leq \left(1 - \frac{\varepsilon}{2}\right) \cdot \Phi_t$$

$$= \underbrace{\boxed{\sum_{i=1}^{n} \omega_t(i)}}_{\Phi_t} - \epsilon \underbrace{\sum_{j \text{ wrong}} \omega_t(j)}_{\geq \frac{1}{2}\Phi_t} \leq \Phi_t \left(1 - \frac{\epsilon}{2}\right)$$

Analysis

Theorem (Multiplicative Weights Update)

Let M_t be the number of mistakes that our algorithm makes until time t, and let $M_t(i)$ be the number of mistakes that expert i made until time t. Then, for any expert $i \in [n]$, we have:

$$M_t \leq 2 \cdot (1+\varepsilon) M_t(i) + \frac{2 \log n}{\varepsilon}$$

- Potential function $\Phi_t = \sum_{i=1}^{n} w_t(i)$
- If we made a mistake, at least half the weight was on the wrong answer. Thus

$$\Phi_{t+1} = \sum_{i \text{ right}} w_t(i) + (1-\varepsilon) \cdot \sum_{j \text{ wrong}} w_t(j) \leq \left(1 - \frac{\varepsilon}{2}\right) \cdot \Phi_t$$

- Thus,

$$\Phi_t = \underbrace{\Phi_1}_{n} \cdot \left(1 - \frac{\varepsilon}{2}\right)^{M_t} = n \cdot \left(1 - \frac{\varepsilon}{2}\right)^{M_t}$$

Analysis

1. We have

$$\Phi_t \leq n \cdot \left(1 - \frac{\varepsilon}{2}\right)^{M_t}$$

Analysis

1. We have

$$\Phi_t \leq n \cdot \left(1 - \frac{\varepsilon}{2}\right)^{M_t}$$

2. On the other hand, have:

$$\Phi_t = \sum_{j=1}^{n} \overbrace{w_t(j)}^{>0} > w_t(i) = (1 - \varepsilon)^{M_t(i)}$$

by definition

multiply weight of i whenever it makes mistake

Analysis

1. We have

$$\Phi_t \leq n \cdot \left(1 - \frac{\varepsilon}{2}\right)^{M_t}$$

2. On the other hand, have:

$$\Phi_t = \sum_{j=1}^{n} w_t(j) > w_t(i) = (1-\varepsilon)^{M_t(i)}$$

3. Putting (1) and (2) together

$$\underbrace{n \cdot \left(1 - \frac{\varepsilon}{2}\right)^{M_t}}_{\geq \Phi_t \quad >} > \underbrace{(1-\varepsilon)^{M_t(i)}}_{\text{weight of } i^{th} \text{ expert}} \xRightarrow{\text{taking logs}} \log(1-\varepsilon/2) \cdot M_t + \log n > M_t(i) \cdot \log(1-\varepsilon)$$

Analysis

$\rightarrow \log\left(1-\frac{\varepsilon}{2}\right) M_t + \log n < -\frac{\varepsilon}{2} M_t + \log n$

$\rightarrow M_t(i) \log(1-\varepsilon) > -(\varepsilon + \varepsilon^2) M_t(i)$

$\Rightarrow -\frac{\varepsilon}{2} M_t + \log n > -(\varepsilon+\varepsilon^2) M_t(i) \Longrightarrow$

$\Rightarrow M_t < \frac{2 \log n}{\varepsilon} + (2+2\varepsilon) M_t(i)$

3. Putting (1) and (2) together

$$n \cdot \left(1-\frac{\varepsilon}{2}\right)^{M_t} > (1-\varepsilon)^{M_t(i)} \Rightarrow \log(1-\varepsilon/2) \cdot M_t + \log n > M_t(i) \cdot \log(1-\varepsilon)$$

4. Using inequality $-x - x^2 < \log(1-x) < -x$ for $x \in (0, 1/2)$, we get:

$$-\varepsilon/2 \cdot M_t + \log n > M_t(i) \cdot (-\varepsilon - \varepsilon^2)$$

Multiplicative Weights Update - General

The same algorithm and argument above, applied to the setting:

Multiplicative Weights Update - General

The same algorithm and argument above, applied to the setting:

1. **Setup:** have access to n experts, indexed by the set $[n]$.

Multiplicative Weights Update - General

The same algorithm and argument above, applied to the setting:

1. **Setup:** have access to n experts, indexed by the set $[n]$.
    - At each time step i, each expert will guess a value $m_t(i) \in [-1, +1]$.
    - Cost of i^{th} expert answer at time t is $\boxed{m_t(i)}$
    - Parameter $0 < \varepsilon < 1/2$

Multiplicative Weights Update - General

The same algorithm and argument above, applied to the setting:

1. **Setup:** have access to n experts, indexed by the set $[n]$.
   - At each time step i, each expert will guess a value $m_t(i) \in [-1, +1]$.
   - Cost of i^{th} expert answer at time t is $m_t(i)$
   - Parameter $0 < \varepsilon < 1/2$
2. Let $p_t : [n] \to \mathbb{R}_+$ weight function (normalized to sum to 1)
   - $p_t(i)$ is the *weight* of expert i at time t
   - In beginning every expert has weight $p_1(i) = 1/n$
   - Our total cost is $\sum_t p_t \cdot m_t$

$$\sum_{i=1}^{n} p_t(i) = 1 \qquad \sum_{j=1}^{n} \frac{w_t(j)}{\sum_{i=1}^{n} w_t(i)} = 1$$

$$\frac{w_t(j)}{\sum_{i=1}^{n} w_t(i)} = p_t(i)$$

$$p_t \cdot m_t = \sum_{i=1}^{n} p_t(i) m_t(i)$$

Multiplicative Weights Update - General

The same algorithm and argument above, applied to the setting:

1. **Setup:** have access to n experts, indexed by the set $[n]$.
    - At each time step i, each expert will guess a value $m_t(i) \in [-1, +1]$.
    - Cost of i^{th} expert answer at time t is $m_t(i)$
    - Parameter $0 < \varepsilon < 1/2$
2. Let $p_t : [n] \to \mathbb{R}_+$ weight function (normalized to sum to 1)
    - $p_t(i)$ is the *weight* of expert i at time t
    - In beginning every expert has weight $p_1(i) = 1/n$
    - Our total cost is $\sum_t p_t \cdot m_t$
3. Our goal is to minimize our total cost: $\boxed{\sum_{t=1}^{T} p_t \cdot m_t}$

Multiplicative Weights Update - General

The same algorithm and argument above, applied to the setting:

1. **Setup:** have access to n experts, indexed by the set $[n]$.
   - At each time step i, each expert will guess a value $m_t(i) \in [-1, +1]$.
   - Cost of i^{th} expert answer at time t is $m_t(i)$
   - Parameter $0 < \varepsilon < 1/2$
2. Let $p_t : [n] \to \mathbb{R}_+$ weight function (normalized to sum to 1)
   - $p_t(i)$ is the *weight* of expert i at time t
   - In beginning every expert has weight $p_1(i) = 1/n$
   - Our total cost is $\sum_t p_t \cdot m_t$
3. Our goal is to minimize our total cost: $\sum_{t=1}^{T} p_t \cdot m_t$

Theorem (Multiplicative Weights Update)

With the setup above, after t rounds, for any expert $i \in [n]$, we have:

our total cost

$$\sum_{t=1}^{T} p_t \cdot m_t \leq \sum_{t=1}^{T} m_t(i) + \varepsilon \cdot \sum_{t=1}^{T} |m_t(i)| + \frac{\ln n}{\varepsilon}$$

- Multiplicative Weights Update
- Solving Linear Programs
- Conclusion
- Acknowledgements

Solving Linear Programs

Assume we are given LP in feasibility version:

$$Ax \geq b$$
$$x \geq 0$$

Solving Linear Programs

Assume we are given LP in feasibility version:

$$
\begin{aligned}
Ax &\geq b \\
x &\geq 0
\end{aligned}
$$

- Optimization version reduces to feasibility version by binary search.

Solving Linear Programs

Assume we are given LP in feasibility version:

$$Ax \geq b$$
$$x \geq 0$$

- Optimization version reduces to feasibility version by binary search.
- Think of $x \geq 0$ being the easy constraints to satisfy, whereas $Ax \geq b$ are the hard ones

Solving Linear Programs

Assume we are given LP in feasibility version:

$$Ax \geq b$$
$$x \geq 0$$

- Optimization version reduces to feasibility version by binary search.
- Think of $x \geq 0$ being the easy constraints to satisfy, whereas $Ax \geq b$ are the hard ones

Idea:

1. Think of each inequality $A_i x \geq b_i$ as an *expert* (A_i is i^{th} row of A)

Solving Linear Programs

Assume we are given LP in feasibility version:

$$Ax \geq b$$
$$x \geq 0$$

- Optimization version reduces to feasibility version by binary search.
- Think of $x \geq 0$ being the easy constraints to satisfy, whereas $Ax \geq b$ are the hard ones

Idea:

1. Think of each inequality $A_i x \geq b_i$ as an *expert* (A_i is i^{th} row of A)
2. Each constraint would like to be *the hardest constraint*, i.e. the one that is violated the most by the current proposed solution $x^{(t)}$

Solving Linear Programs

Assume we are given LP in feasibility version:

$$Ax \geq b$$
$$x \geq 0$$

- Optimization version reduces to feasibility version by binary search.
- Think of $x \geq 0$ being the easy constraints to satisfy, whereas $Ax \geq b$ are the hard ones

Idea:

1. Think of each inequality $A_i x \geq b_i$ as an *expert* (A_i is i^{th} row of A)
2. Each constraint would like to be *the hardest constraint*, i.e. the one that is violated the most by the current proposed solution $x^{(t)}$
3. More precisely: cost of i^{th} constraint

$$\boxed{A_i x - b_i}$$

Solving Linear Programs

Assume we are given LP in feasibility version:

$$Ax \geq b$$
$$x \geq 0$$

- Optimization version reduces to feasibility version by binary search.
- Think of $x \geq 0$ being the easy constraints to satisfy, whereas $Ax \geq b$ are the hard ones

Idea:

1. Think of each inequality $A_i x \geq b_i$ as an *expert* (A_i is i^{th} row of A)
2. Each constraint would like to be *the hardest constraint*, i.e. the one that is violated the most by the current proposed solution $x^{(t)}$
3. More precisely: cost of i^{th} constraint

$$A_i x - b_i$$

4. We would like to propose feasible solution (i.e. lower cost of *all constraints*). Hard to deal with all constraints at the same time.

Solving Linear Programs

- Would like to minimize

$$\boxed{\min_{1 \le i \le m} A_i x - b_i}$$

- Multiplicative Weights Update (MWU) provides way of combining *all constraints* into *one constraint*!

Solving Linear Programs

- Would like to minimize

$$\min_{1 \leq i \leq m} A_i x - b_i$$

- Multiplicative Weights Update (MWU) provides way of combining *all constraints* into *one constraint*!
- MWU finds probability distribution over experts (normalized weights), which in our case are the inequalities.

$p^{(t)}$

$p_i^{(t)}$

weight of
i^{th} expert

Solving Linear Programs

- Would like to minimize

$$\min_{1 \le i \le m} A_i x - b_i$$

- Multiplicative Weights Update (MWU) provides way of combining *all constraints* into *one constraint*!
- MWU finds probability distribution over experts (normalized weights), which in our case are the inequalities.
- Thus, we have to deal with only the constraint $\boxed{p^{(t)} A x \geq p^{(t)} b}$, where

$p^{(t)} \geq 0$

$$p^{(t)} = \frac{1}{\sum_i w_t(i)} \cdot (w_t(1), \ldots, w_t(n))$$

normalized weights

$x^{(t)} \geq 0$ and satisfies constraint

$Ax \geq b$

$\Rightarrow p^{(t)} A x \geq p^{(t)} b$

Solving Linear Programs

- Would like to minimize

$$\min_{1 \le i \le m} A_i x - b_i$$

- Multiplicative Weights Update (MWU) provides way of combining *all constraints* into *one constraint*!
- MWU finds probability distribution over experts (normalized weights), which in our case are the inequalities.
- Thus, we have to deal with only the constraint $p^{(t)} A x \ge p^{(t)} b$, where

$$p^{(t)} = \frac{1}{\sum_i w_t(i)} \cdot (w_t(1), \ldots, w_t(n))$$

- MWU shows that over the long run:

The *total violation* of our *weighted constraints* will be close to the *total violation* of the *worst violated constraint*!

Solving Linear Programs

- Would like to minimize

$$\min_{1 \leq i \leq m} A_i x - b_i$$

Solving Linear Programs

- Would like to minimize

$$\min_{1 \le i \le m} A_i x - b_i$$

- MWU shows that over the long run, for any $\underbrace{\text{inequality } i}_{\text{expert } i} \in [m]$:

$$\underbrace{\sum_{t=1}^{T} \underbrace{p^{(t)} \cdot (Ax^{(t)} - b)}_{\text{our cost at } t}}_{\text{total cost}} < \underbrace{\frac{\log m}{\varepsilon}}_{m \text{ inequalities}} + \sum_{t=1}^{T} \underbrace{(A_i x^{(t)} - b_i)}_{\text{cost of } i^{\text{th}} \text{ expert at time } t} + \varepsilon \cdot \sum_{t=1}^{T} |A_i x^{(t)} - b_i|$$

$A_i x - b_i$

$\hookrightarrow A_i x^{(t)} - b_i$

Solving Linear Programs

- Would like to minimize

$$\min_{1 \le i \le m} A_i x - b_i$$

- MWU shows that over the long run, for any inequality $i \in [m]$:

$$\sum_{t=1}^{T} p^{(t)} \cdot (Ax^{(t)} - b) < \frac{\log m}{\varepsilon} + \sum_{t=1}^{T} (A_i x^{(t)} - b_i) + \varepsilon \cdot \sum_{t=1}^{T} |A_i x^{(t)} - b_i|$$

- But our theorem required $m_t(i) \in [-1, +1]$... $\underbrace{\text{How can we fix this?}}_{\text{see later slides}}$

$m_t(i) = A_i x^{(t)} - b_i$

Solving Linear Programs

- Would like to minimize

$$\min_{1 \le i \le m} A_i x - b_i$$

- MWU shows that over the long run, for any inequality $i \in [m]$:

$$(1) \quad \sum_{t=1}^{T} p^{(t)} \cdot (Ax^{(t)} - b) < \frac{\log m}{\varepsilon} + \sum_{t=1}^{T} (A_i x^{(t)} - b_i) + \varepsilon \cdot \sum_{t=1}^{T} |A_i x^{(t)} - b_i|$$

- But our theorem required $m_t(i) \in [-1, +1]$... How can we fix this?
- Return solution

$$\boxed{x = \frac{1}{T} \cdot \sum_{t=1}^{T} x^{(t)}}$$

$$\frac{1}{T} \sum_{t=1}^{T} (A_i x^{(t)} - b_i) = A_i \cdot \underbrace{\frac{1}{T} \sum_t x^{(t)}}_{x} - b = A_i x - b_i$$

Solving Linear Programs

$p_i^{(t)} \geq 0$

- Would like to minimize

$$\min_{1 \le i \le m} A_i x - b_i$$

- MWU shows that over the long run, for any inequality $i \in [m]$:

$$\sum_{t=1}^{T} p^{(t)} \cdot (Ax^{(t)} - b) < \frac{\log m}{\varepsilon} + \sum_{t=1}^{T} (A_i x^{(t)} - b_i) + \varepsilon \cdot \sum_{t=1}^{T} |A_i x^{(t)} - b_i|$$

- But our theorem required $m_t(i) \in [-1, +1]$... How can we fix this?
- Return solution

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

- What if ~~we cannot find~~ (if there isn't one) any $x \geq 0$ such that $\boxed{p^{(t)} Ax \geq p^{(t)} b}$?
  - Farkas' lemma $\Rightarrow$ the system is *infeasible*, and we are done!

note that $p^{(t)} A x \geq p^{(t)} b$ is a convex combination of inequalities $Ax \geq b$

Solving Linear Programs

- Would like to minimize

$$\min_{1 \le i \le m} A_i x - b_i$$

- MWU shows that over the long run, for any inequality $i \in [m]$:

$$\sum_{t=1}^{T} p^{(t)} \cdot (Ax^{(t)} - b) < \frac{\log m}{\varepsilon} + \sum_{t=1}^{T} (A_i x^{(t)} - b_i) + \varepsilon \cdot \sum_{t=1}^{T} |A_i x^{(t)} - b_i|$$

- But our theorem required $m_t(i) \in [-1, +1]$... How can we fix this?
- Return solution

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

- What if we cannot find any $x \ge 0$ such that $p^{(t)} Ax \ge p^{(t)} b$?
  - Farkas' lemma $\Rightarrow$ the system is *infeasible*, and we are done!
  - Thus, we will assume that above never happens.

Theorem

Definition (Oracle)

Let $A \in \mathbb{R}^{m \times n}$. We say that $\mathcal{O}$ is an oracle of *width w* for A if given a linear constraint

$$pAx \geq pb, \;\; x \geq 0$$

$\mathcal{O}(p)$ will return $\boxed{y \geq 0}$ such that $pAy \geq pb$

$$\boxed{|A_i y - b_i| \leq w} \quad \forall i \in [m]$$

Moreover, if there is no solution to the linear constraint, then $\mathcal{O}$ will return NONE

Theorem

Definition (Oracle)

Let $A \in \mathbb{R}^{m \times n}$. We say that $\mathcal{O}$ is an oracle of *width w* for A if given a linear constraint

$$pAx \geq pb, \;\; x \geq 0$$

$\mathcal{O}(p)$ will return $y \geq 0$ such that

$$|A_i y - b_i| \leq w \quad \forall i \in [m]$$

Theorem (Multiplicative Weights Update)

Let $\delta > 0$ and suppose we are given an oracle with width w for A. The MWU algorithm either finds a solution $y \geq 0$ such that

error parameter

$$\boxed{A_i y \geq b_i - \delta \quad \forall i \in [m]}$$

or concludes that the system is infeasible (and outputs a dual solution). Our algorithm makes $O(w^2 \log(m)/\delta^2)$ oracle calls.

Analysis

- As we said before, if oracle fails to find a solution, we found a separating hyperplane and we are done.

Analysis

$p^{(t)}(Ax^{(t)}-b) \geq 0 \qquad x^{(t)} \geq 0$

- As we said before, if oracle fails to find a solution, we found a separating hyperplane and we are done.
- Otherwise, we have that MWU algorithm with costs

$$m_t(i) = \boxed{\frac{A_i x^{(t)} - b_i}{w}} \in [-1,+1]$$

gives us that after T steps

$$0 \leq \underbrace{\sum_{t=1}^{T} p^{(t)} \cdot \frac{A_i x^{(t)} - b_i}{w}}_{\text{our cost at time } t} < \frac{\log m}{\varepsilon} + \underbrace{\sum_{t=1}^{T} \frac{A_i x^{(t)} - b_i}{w}}_{i\text{th experts cost}} + \varepsilon \cdot \overbrace{\sum_{t=1}^{T} \underbrace{\frac{|A_i x^{(t)} - b_i|}{w}}_{\leq 1}}^{\leq T}$$

≥ 0

as our oracle gives solution to

$p^{(t)}(A_i x^{(t)} - b) \geq 0 \quad \therefore$ sum also ≥ 0

as we have by oracle that

$|A_i x^{(t)} - b_i| \leq w$

$$0 \leq \sum_{i=1}^{t} \frac{p^{(t)}(Ax^{(t)}-b)}{w} < \frac{\log m}{\varepsilon} + \sum_{i=1}^{T} \frac{A_i x^{(t)} - b_i}{w} + \varepsilon T$$

Analysis

- As we said before, if oracle fails to find a solution, we found a separating hyperplane and we are done.
- Otherwise, we have that MWU algorithm with costs $m_t(i) = \frac{A_i x^{(t)} - b_i}{w}$ gives us that after T steps

$$0 \leq \sum_{t=1}^{T} p^{(t)} \cdot \frac{A x^{(t)} - b}{w} < \frac{\log m}{\varepsilon} + \sum_{t=1}^{T} \frac{A_i x^{(t)} - b_i}{w} + \varepsilon \cdot \sum_{t=1}^{T} \frac{|A_i x^{(t)} - b_i|}{w}$$

T

- Thus, we have

$$\boxed{\sum_{t=1}^{T} \frac{A_i x^{(t)} - b_i}{T}} \geq - \frac{w \log m}{T \cdot \varepsilon} - \varepsilon \cdot w$$

$$0 \leq \frac{\log m}{\varepsilon} + \boxed{\sum_{t=1}^{T} \frac{A_i x^{(t)} - b_i}{w}} + \varepsilon \cdot T$$

Analysis

- As we said before, if oracle fails to find a solution, we found a separating hyperplane and we are done.
- Otherwise, we have that MWU algorithm with costs $m_t(i) = \frac{A_i x^{(t)} - b_i}{w}$ gives us that after T steps

$$\sum_{t=1}^{T} p^{(t)} \cdot \frac{A_i x^{(t)} - b_i}{w} < \frac{\log m}{\varepsilon} + \sum_{t=1}^{T} \frac{A_i x^{(t)} - b_i}{w} + \varepsilon \cdot \sum_{t=1}^{T} \frac{|A_i x^{(t)} - b_i|}{w}$$

- Thus, we have

$$\sum_{t=1}^{T} \frac{A_i x^{(t)} - b_i}{T} \geq -\frac{w \log m}{T \cdot \varepsilon} - \varepsilon \cdot w$$

- Setting $\varepsilon = \delta/2w$ and $T = \frac{4 \cdot w^2 \cdot \log m}{\delta^2}$ we get

$$A_i x - b_i = \sum_{t=1}^{T} \frac{A_i x^{(t)} - b_i}{T} \geq -\delta$$

Conclusion

- Game Theory
- Online Learning
  1. Experts are weak classifiers, want to choose hypothesis based on these experts
  2. Boosting (in learning theory)
- Solving linear programs! (today)
- Convex Optimization
- Computational Geometry
- many more

Acknowledgement

- Lecture based largely on:
  - Lap Chi's notes
  - Yaron Singer's notes
  - Elad Hazan's survey on online optimization
- See Lap Chi's notes at `https://cs.uwaterloo.ca/~lapchi/cs466/notes/L21.pdf`
- See Yaron's notes `https://people.seas.harvard.edu/~yaron/AM221-S16/lecture_notes/AM221_lecture11.pdf`
- See Elad's survey at `https://arxiv.org/pdf/1909.05207.pdf`
- See great survey on MWU ar `https://www.cs.princeton.edu/~arora/pubs/MWsurvey.pdf`