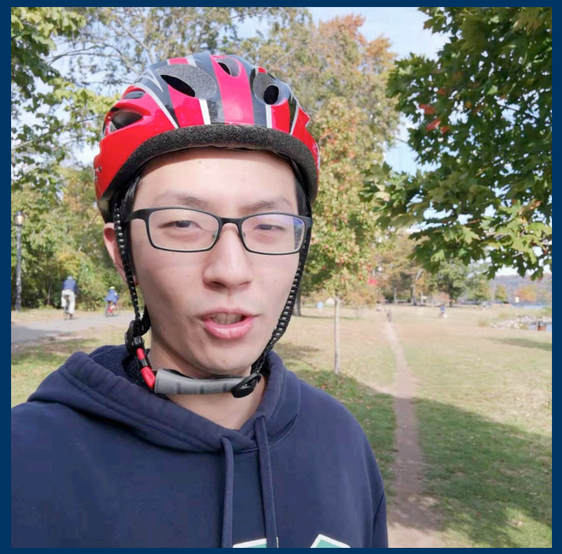
Online Prediction in Sub-linear Space

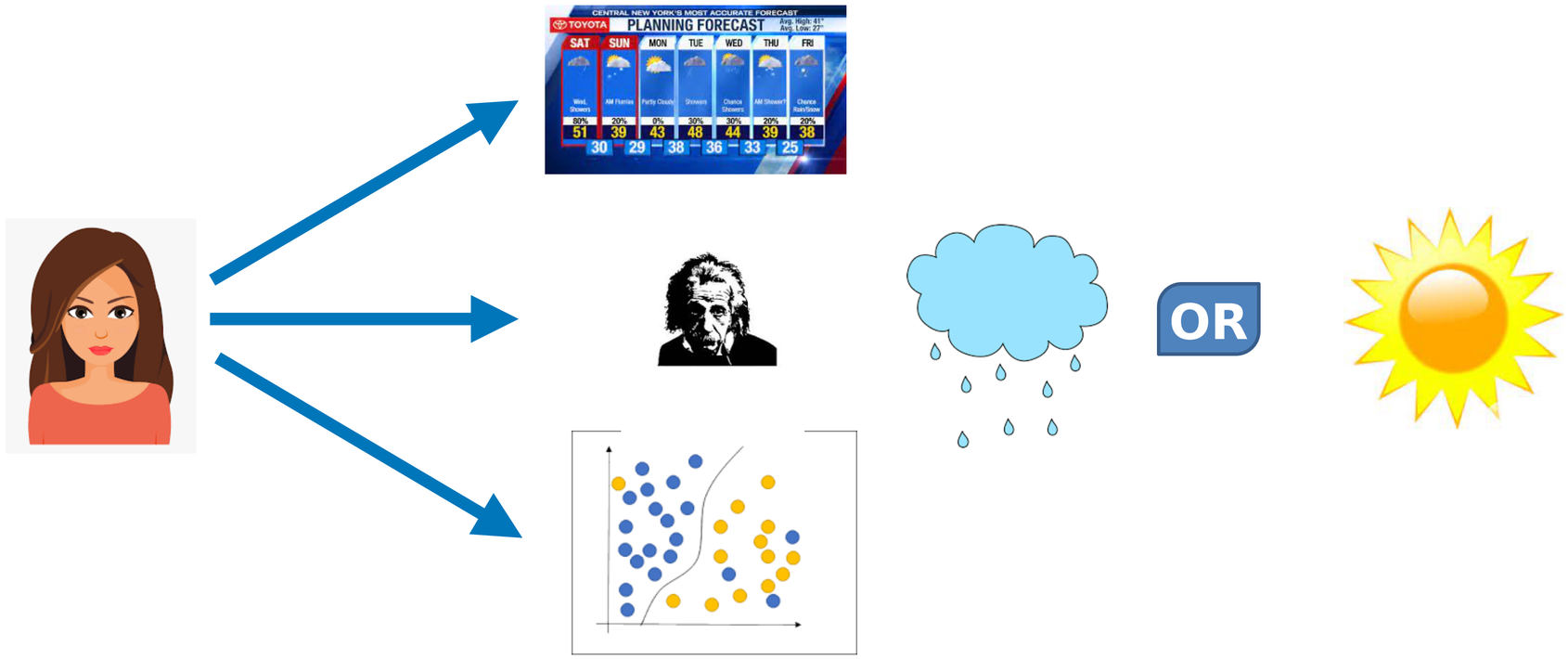
Binghui Peng (Columbia)



Fred Zhang (UC Berkeley)

Forecasting

Online and over many days



High Level Pitch

Online learning meets streaming algorithm

- The experts problem
  - The most basic question in online learning
- Classically: multiplicative weights update (MWU) method
  - Optimal $\sqrt{T}$ regret, but $\Omega(n)$ space complexity
  - **Main result**: first algorithm with small regret & sub-linear memory
  - **Conceptually**: a sub-linear space version of MWU
- Opens up many new directions (more soon!)

Outline

- Background & problem settings
- A simple hardness result
- Our results
- Algorithms & techniques
- Open directions

Outline

- Background & problem settings
- A simple hardness result
- Our results
- Algorithms & techniques
- Open directions

Online Prediction with Experts Advice

Formalizing weather forecasting

- An unknown sequence {*0, 1*} of *T* days
- Experts from *1* to *n*
  - At day *t*, expert *i* recommends $x_t(i) \in \{0,1\}$
- Algorithm follows one of them & suffers loss *1* iff it makes mistake

---

$$\text{Regret}(T) = \sum_{t \in [T]} \ell_t(\text{ALG}) - \min_{i^* \in [n]} \sum_{t \in [T]} \ell_t(i^*)$$

Online Learning with Experts Advice

A slight generalization and better known

- *T* days, *n* experts
- No sequence to predict
- Algorithm *plays* one of the experts from [*n*]
- At the end of the day, each expert receives a loss $\ell_t(i) \in [0,1]$

$$\text{Regret}(T) = \sum_{t \in [T]} \ell_t(\text{ALG}) - \min_{i^* \in [n]} \sum_{t \in [T]} \ell_t(i^*)$$

Adversary Model?

Three variants

i) **Non-adaptive**: the loss vectors $\ell_t \in [0,1]^n$ are fixed upfront

ii) **Blackbox adaptive**: the adversary sees the output of the algorithm and may adapt future loss values

iii) **White-box adaptive**: the adversary may even see the internal state of the algorithm

This talk: focus on non-adaptive model, the most classic in literature

Goal

Regret minimization

- **Known**: can achieve $\sqrt{T \log n}$ regret
- Many algorithms can do this
  - Naive (Follow-the-Leader): play the expert with minimum historical loss
  - Known: regret = $\Omega(T)$
  - Classic algorithms refine this idea

How to Solve it?

i) Multiplicative Weights Update

✤ Algorithm [MWU]:

- Initialize weight *w = (1/n, 1/n, …, 1/n)*
- For each day *t*:
  - *Play*: expert i with probability proportional to w_i
  - *Observe*: loss $\ell_t(i)$, for each $i \in [n]$
  - *Update*: $w_i \leftarrow w_i(1 - \ell_t(i)/2)$

Comments on MWU

- **Equivalence**: Follow-the-Regularized-Leader (FTRL)

❖ Algorithm [FTRL]

$$\textit{Update}: p(t) = \arg\min_{p \in \Delta_n} \left(\sum_{j \le t} \langle p, \ell_j \rangle + \eta \cdot \psi(p) \right)$$

- **Regret**: $\sqrt{T}$
- **Memory size**: $\Omega(n)$

How to Solve it?

ii) Follow-the-Perturbed-Leader (FTPL)

❖ Algorithm [FTPL]

Play: $i \in \arg\min_i \left(\sum_{j<t} \ell_j(i) + \varepsilon_i \right)$

- **Regret**: $\sqrt{T}$
- **Memory size**: $\Omega(n)$

Take-away Message

- **Goal**: design o(n) space algorithm with small o(T) regret
- **Main barrier**: leader selection
  - All classic algorithms follow this paradigm
  - We have to get around it

Outline

- [x] Background & problem settings
- [ ] A simple hardness result
- [ ] Our results
- [ ] Algorithms & techniques
- [ ] Open directions

What can we do then?

Find out the best expert in small space?

- Let's say $T \to \infty$
  - Once I know the best expert, keep playing it and suffer 0 regret forever
- Theorem [No-Go against Best Expert Identification]:

  Identifying best expert cannot be done in o(*n*) space

Srinivas, Woodruff, Xu & Zhou, STOC 22

Proof

Via communication complexity reduction

- Set Disjointness —needs $\Omega(n)$ bits of communication
  - Alice holds $X \in \{0,1\}^n$; Bob holds $Y \in \{0,1\}^n$
  - Promise problem
  - Distinguish $|X \cap Y| = 0$ or 1

Proof

Continued

- Reduction
  - Alice: a stream of $T = n$ days. Expert i is correct on day i iff $X_i = 1$
  - Bob: similarly
  - Combine the stream —> instance of online learning with $2n$ days

$$X = 11001 = \{1,2,5\}$$

	Day 1	Day 2	Day 3	Day 4	Day 5
Expert 1	✅	❌	❌	❌	❌
Expert 2	❌	✅	❌	❌	❌
Expert 3	❌	❌	❌	❌	❌
Expert 4	❌	❌	❌	❌	❌
Expert 5	❌	❌	❌	❌	✅

Proof

Disjoint case

$X = 11001 = \{1,2,5\}, Y = \{3,4\}$

All experts are correct at most once

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
Expert 1	✅	❌	❌	❌	❌	❌	❌	❌	❌	❌
Expert 2	❌	✅	❌	❌	❌	❌	❌	❌	❌	❌
Expert 3	❌	❌	❌	❌	❌	❌	❌	✅	❌	❌
Expert 4	❌	❌	❌	❌	❌	❌	❌	❌	✅	❌
Expert 5	❌	❌	❌	❌	✅	❌	❌	❌	❌	❌

Proof

Non-disjoint case

$X = 11001 = \{1,2,5\}, Y = \{1,3,4\}$

Expert *1* is correct twice

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
Expert 1	✅	❌	❌	❌	❌	✅	❌	❌	❌	❌
Expert 2	❌	✅	❌	❌	❌	❌	❌	❌	❌	❌
Expert 3	❌	❌	❌	❌	❌	❌	❌	✅	❌	❌
Expert 4	❌	❌	❌	❌	❌	❌	❌	❌	✅	❌
Expert 5	❌	❌	❌	❌	✅	❌	❌	❌	❌	❌

Proof

Finishing off

- Create the instance by the reduction
- Alice runs the best arm identification algorithm, sends her memory to Bob, and Bob continues —> best expert
  - The best expert is correct on exact 2 or <= 1 days
  - Corresponding to the two cases of Set Disjointness
- Run another round of communication to double check
- **Conceptual message**: best arm identification harder than regret minimization in sub-linear space

Outline

- [x] Background & problem settings
- [x] A simple hardness result
- [ ] Our results
- [ ] Algorithms & techniques
- [ ] Open directions

Main Result

Think of $T \to \infty$

❖ Theorem [Peng & Zhang, SODA '23]

An online learning algorithm (under non-adaptive adversary) that

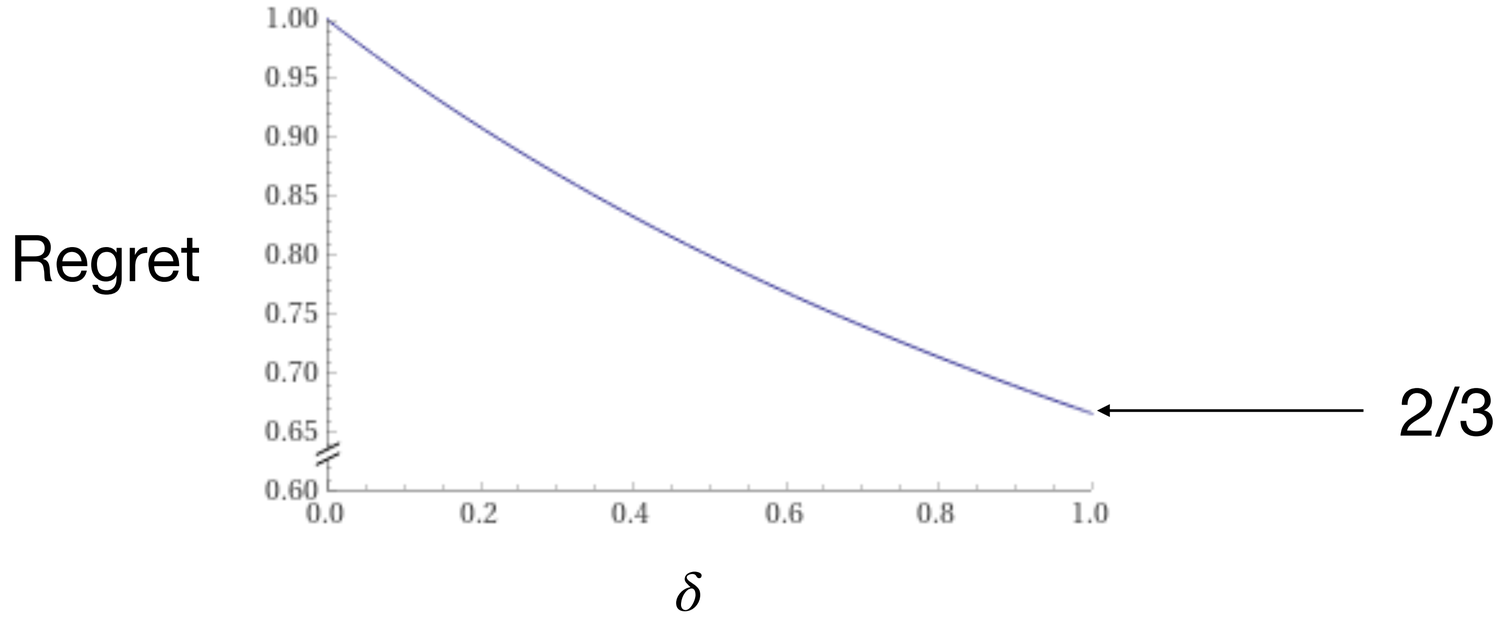
- uses n^{δ} memory;
- gets a total regret of $O\left(T^{2/(2+\delta)}n^2\right)$

Examples:

- $n^{0.99}$ memory & $T^{2/3}$ regret (with $\delta = 0.99$)
- $\sqrt{n}$ memory & $T^{4/5}$ regret (with $\delta = 0.5$)

A Memory-Regret Trade-off

Upper bound



This turns out to be far from tight!

Known Lower Bound

Prior work by Srinivas, Woodruff, Xu & Zhou, STOC 22

- Theorem [Memory *Lower Bound* for Experts Learning]

  For any algorithm with S bits of memory, regret is at least $\Omega\left(\sqrt{nT/S}\right)$

- Examples:
  - With $S = n$, regret is $\Omega\left(\sqrt{T}\right)$
  - With $S = \sqrt{n}$, regret is $\Omega\left(n^{1/4}\sqrt{T}\right)$
  - With $S = O(1)$, regret is $\Omega\left(\sqrt{nT}\right)$

Follow-up Work

Peng & Rubinstein (arXiv, 2023)

- $O_n\left(\sqrt{T}\right)$ regret is possible, even in constant space!
- Theorem [Tight Memory-Regret Tradeoff for Online Learning]

  An algorithm matching the lower bound (up to polylogs)
- **Same framework of this talk** — though a quite a tour de force

Outline

- [x] Background & problem settings
- [x] A simple hardness result
- [x] Our results
- [ ] Algorithms & techniques
- [ ] Open directions

Overview

A baseline algorithm + bootstrapping

i) A baseline algorithm

- Regret: εT regret
- Memory: $1/\varepsilon^2$ space

ii) A hierarchical “width reduction” procedure

- Regret: From εT to $T^{1-\delta(\varepsilon)}$
- Memory: small blow-up

A Natural Idea

Subsampling!

❖ Baseline Algorithm [A High-Level Abstraction]

- Maintain a small subset of experts (*the pool*)
- Run MWU on them

Intuition: Reduce regret minimization to pooling good experts
(to be formalized)

Baseline Algorithm

Break the *T* days into epochs of length *B*

Very simple

❖ Algorithm [Baseline]

- Initially, sample an arbitrary pool of $1/\varepsilon^2$ experts
- Play MWU(Pool) everyday
- Within each epoch:
  - At the beginning, sample *1* expert into the pool (reinitialize the MWU)
  - At the end, apply **some eviction rule** to remove experts

Naive Attempt

Just look at the average loss?

- Evict the worst-performing expert?
- Counter-example:
  - One best expert, good on average
  - Many bad experts, but they excel locally (on small regions)
  - Best expert can be evicted and hard to come back

The Lesson

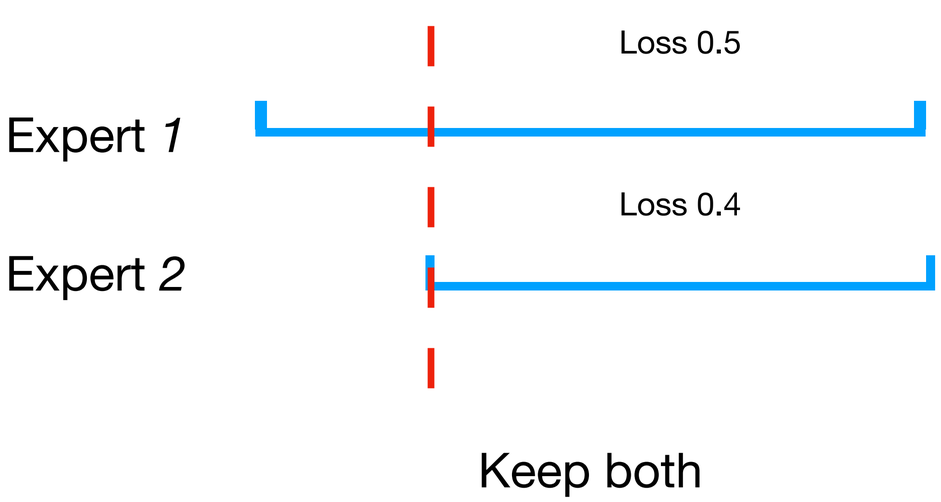
We need stability

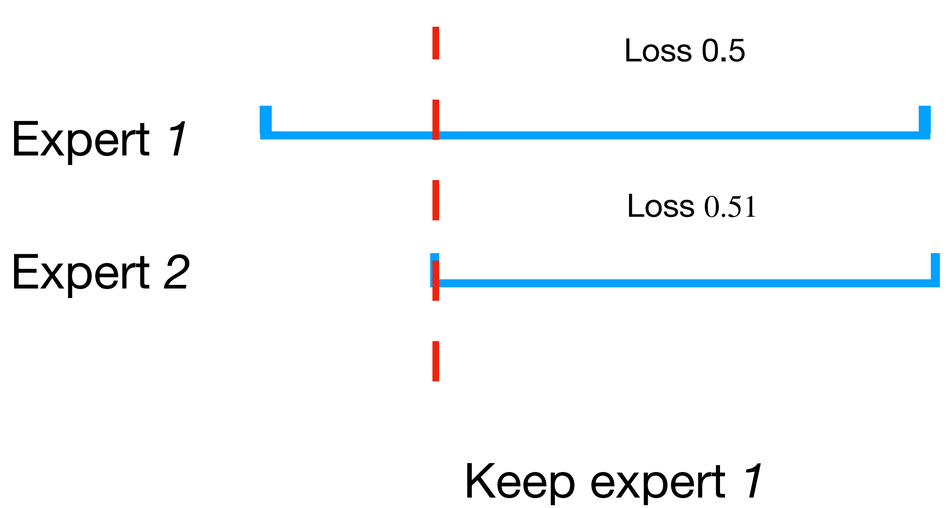
$$\text{Average Loss of } i = \frac{\text{Cumulative Loss of } i}{\text{Age of } i}$$

- Non-robust with respect to **local** performance
- Average performance is useful—just need to be looked at differently
  - For a old expert, its average loss is “stable”
  - For a young expert, it’s not
- **Key idea**: Respect senior folks! This stabilizes the algorithm
  - Keep good experts
  - Keep old experts

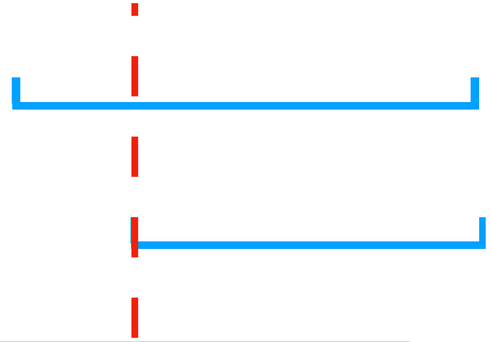
Baseline Algorithm: Eviction Rule

- **High level**: pairwise tournament, any expert that gets dominated is evicted
- Definition [dominance] An expert *i dominates* an expert *j*, if
  - i) Expert *i* is older than expert *j*; and
  - ii) Over *j*'s lifetime, expert *i*'s average loss <= *j's*





Baseline Algorithm: Eviction Rule



Refined

Let $L_j(i)$ = average loss of *i* over *j*'s lifetime

❖ Definition [dominance]. An expert *i dominates* an expert *j*, if

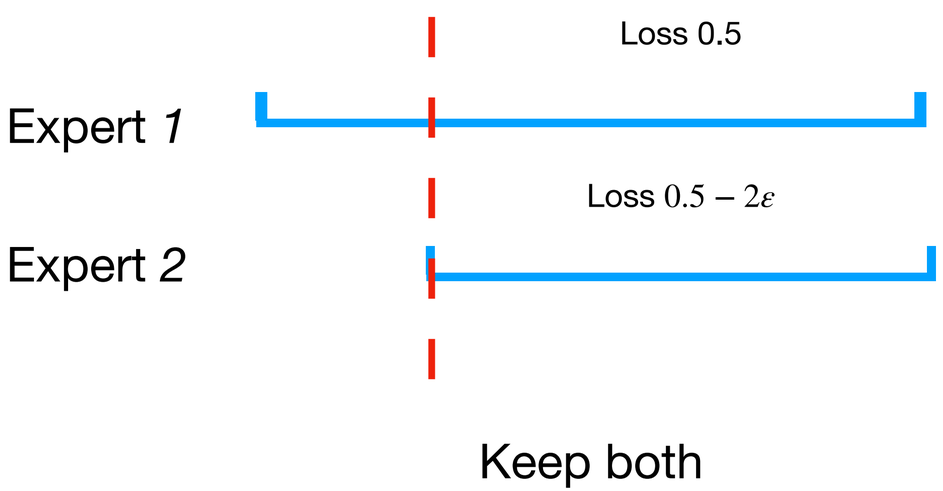
i) Expert *i* is older than expert *j*; and

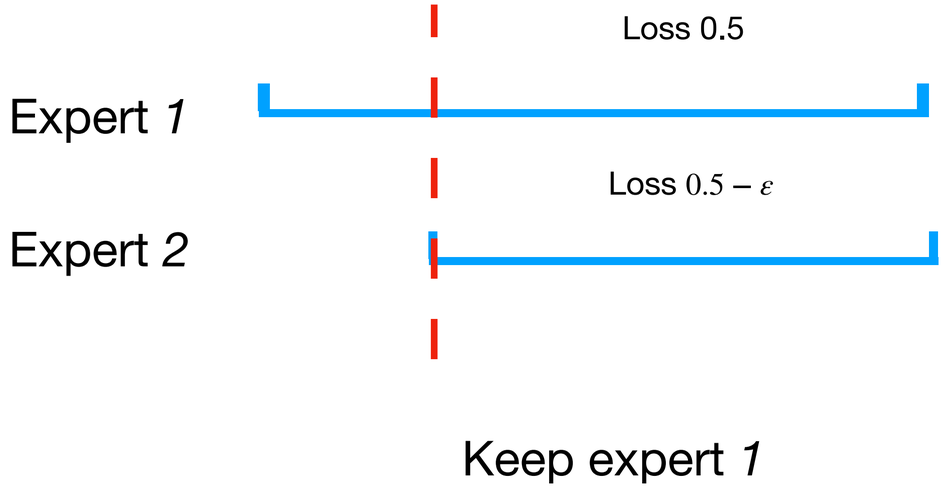
ii) $L_j(i) \leq L_j(j) + \varepsilon$

Interpretation: for *j* to survive, it has to be better than *i* by an ε margin

Baseline Algorithm: Eviction Rule

In pictures





Next: memory analysis, then regret

Memory Analysis

Bound pool size by $1/\varepsilon^2$

❖ Lemma [loss vs length]

Let P_t = the pool at time *t*.

Let β_i = the lifetime of *i* for $i \in P_t$. Let $L_j(i)$ = average loss of *i* over *j*'s lifetime.

After eviction, take *i* older than *j* ($\beta_i \geq \beta_j$). Then either

i) $\beta_i \geq (1+\varepsilon)\beta_j$; or

ii) $L_i(i) \geq L_j(j) + \varepsilon/2$

Memory Analysis

Picture for the key lemma

i) Either $\beta_i \geq (1+\varepsilon)\beta_j$; or

ii) $L_i(i) \geq L_j(j) + \varepsilon/2$

Key Lemma

Proof Sketch

i) Either $\beta_i \geq (1+\varepsilon)\beta_j$; or

ii) $L_i(i) \geq L_j(j) + \varepsilon/2$

- Assume ii) doesn't hold, and show i) has to hold

Length $\geq (1+\epsilon)L$

Length = L

OR

loss $\geq \ell + \varepsilon/2$

loss = ℓ

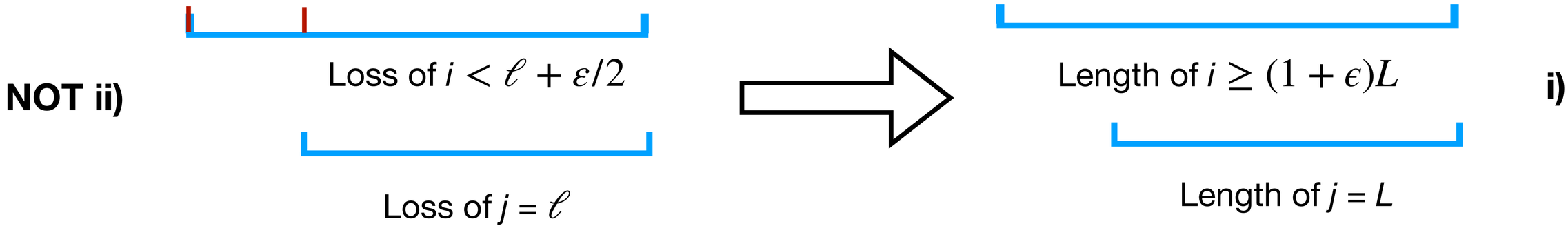
Key Lemma

Proof Sketch

i) Either $\beta_i \geq (1+\varepsilon)\beta_j$; or

ii) $L_i(i) \geq L_j(j) + \varepsilon/2$

- Assume ii) doesn't hold. Show i) has to hold
  - The eviction rule => $L_j(i) > L_j(j) + \varepsilon$
  - By assumption => $L_i(i) < L_j(j) + \varepsilon/2$
  - Loss of *i* in red segment is small + loss is [0,1] => red segment is long



Memory Analysis

Pool size bound

i) Either $\beta_i \geq (1+\varepsilon)\beta_j$; or

ii) $L_i(i) \geq L_j(j) + \varepsilon/2$

- Order experts in pool by their age: 1 -> 2 -> 3 -> 4 ->…
- Naive argument: consider adjacent pairs
  - If always case i), cannot happen for > $\frac{\log T}{\log(1+\varepsilon)} = O(\log T/\varepsilon)$ times
  - If always case ii), cannot happen for > $1/\varepsilon$ times

Length $\geq (1+\epsilon)L$

Length = L

OR

loss $\geq \ell + \varepsilon/2$

loss = ℓ

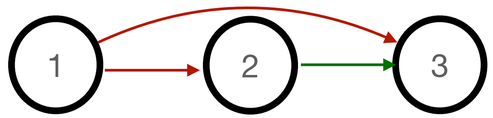
Memory Analysis

Pool size bound

i) Either $\beta_i \geq (1+\varepsilon)\beta_j$; or

ii) $L_i(i) \geq L_j(j) + \varepsilon/2$

- Claim: Pool size $\leq 1/\varepsilon^2$
- Between any *i* older than *j* in the pool
  - *i* —> *j* if $\beta_i \geq (1+\varepsilon)\beta_j$ (length ↓)
  - *i* —> *j* if $L_i(i) \geq L_j(j) + \varepsilon/2$ (loss ↓)
- A DAG with edge colors
  - Longest green chain $\leq \log T/\varepsilon$; and longest red chain $\leq 1/\varepsilon$
  - Size of graph $\leq 1/\varepsilon^2$ (can be formalized via Dilworth's theorem)



Remarks

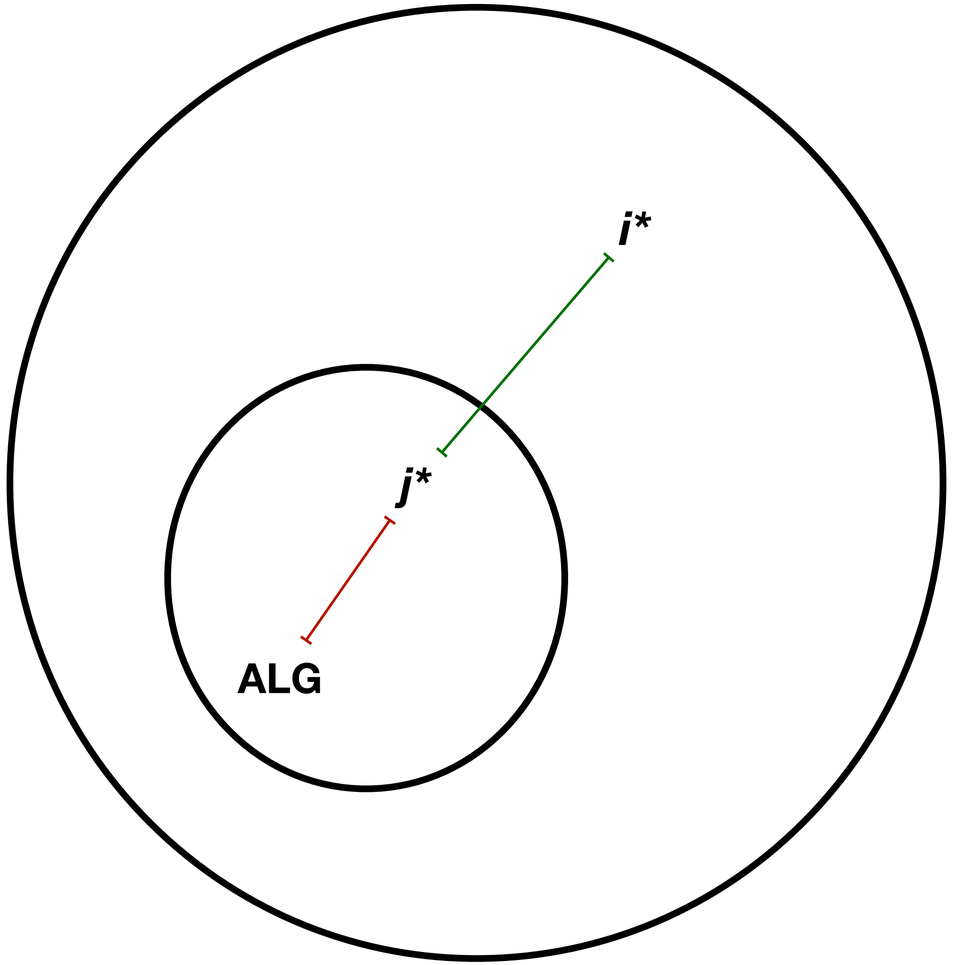
Finishing off memory analysis

- Recall: eviction rule makes pairwise comparisons **on sub-intervals**
  - Quadratic blow up from pool size to actual memory usage
  - $=>1/\varepsilon^4$ memory
- A potential function argument to bound the pool size by $1/\varepsilon$
  - $=>1/\varepsilon^2$ memory as promised
  - See the paper for details

Regret Analysis

At a high level

- Regret = “Inner-Regret” + “Outer-Regret”
  - “Inner-Regret” = ALG - best expert in the pool
  - “Outer-Regret” = best in pool (*j**) - best overall (*i**)
- Inner-Regret is small by MWU
- Outer-Regret is small if the pool contains *good* experts in general



Regret Analysis

Epoch by epoch

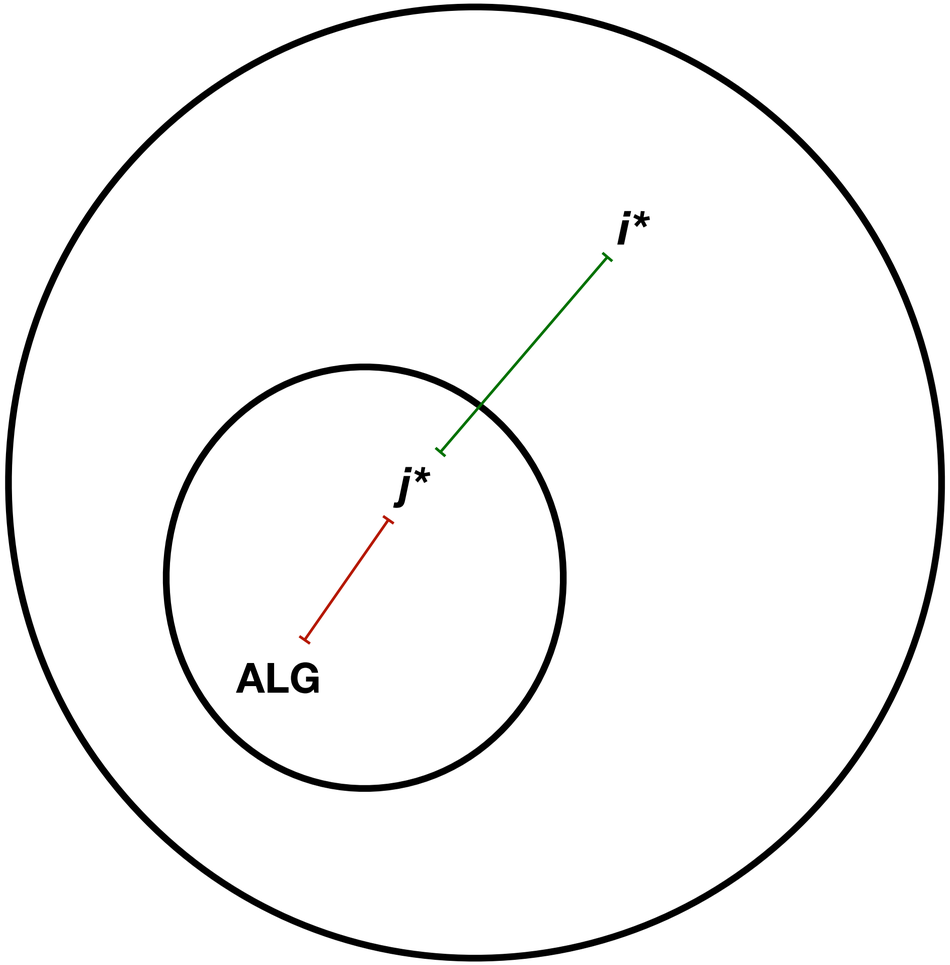
Epoch length = B

Best expert = i^*

Regret = "Inner-Regret" + "Outer-Regret"

i) Inner-Regret = $\underbrace{(T/B)}_{\text{\# Epochs}} \cdot \underbrace{\sqrt{B}}_{\text{Regret per epoch}}$

ii) Outer-Regret: a thought experiment



Analysis of Outer-Regret

Thought experiment

Epoch length = B

Best expert = i^*

- Recall at the beginning of each epoch, we sample *1* expert into the pool
- Consider a fixed epoch
- **Thought experiment**: imagine i^* had been sampled at its beginning (regardless what happened in the actual execution of the algorithm)
- Definition [Evict and stay epoch]
  - An epoch is an evict epoch if i^* would be evicted eventually (at some point in future)
  - An epoch is an stay epoch if i^* would stay forever

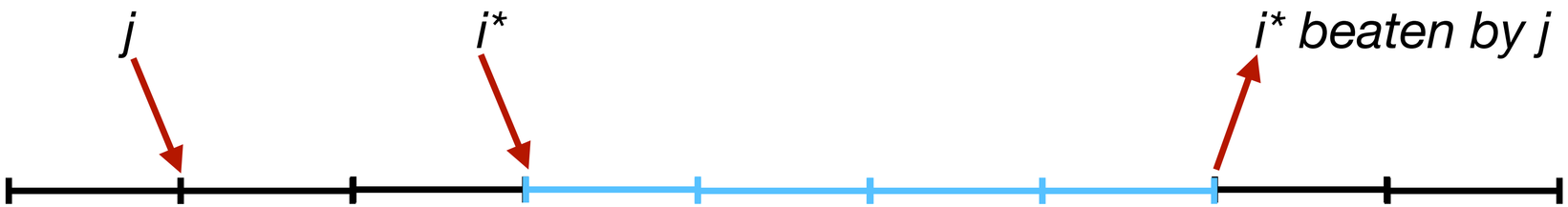
Evict Epoch

They are good

Epoch length = B

Best expert = i^*

- **Intuition**: evict epochs are good epochs
- If i^* is evicted by some j, then due to the eviction rule
  - j is an older expert
  - j's average loss is ~smaller than i over i^*'s lifetime, up to ε
- => Within i^*'s lifetime, j's loss is only ε worse than i^* per day (on average)
- => Within i^*'s lifetime, outer-regret per day is at most ε

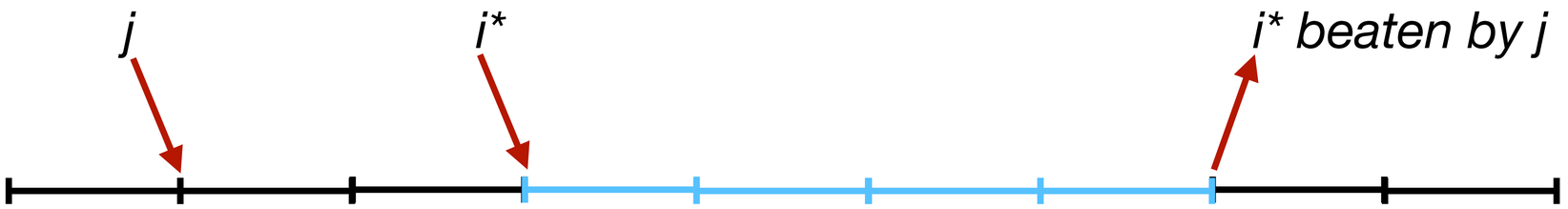


Evict Epoch

Epoch length = B

Best expert = i^*

- **Conclusion**: Within i^*'s lifetime, average outer-regret is at most ε
- Potentially, all epochs evict epochs
  - In total, outer-regret, due to evict epochs, is $\leq \varepsilon T$



Stay Epoch

They are bad but can be bounded

Epoch length = B

Best expert = i^*

- **Intuition**: stay epochs are bad epochs
- No expert in pool can compete with i^*
- Key idea: bound the number of stay epochs
  - If in one of the stay epochs, i^* actually got sampled, we are done!
  - Foe each stay epoch, this happens with probability $1/n$
  - After $O(n \log n)$ stay epochs, i^* would be sampled into the pool with high probability

Stay Epoch

Epoch length = B

Best expert = i^*

- **Conclusion**: at most $O(n \log n)$ stay epochs
- Naively, outer-regret within each stay epoch $\leq B$
  - => In total, total outer-regret, due to stay epochs, is $\leq Bn \log n$

Finishing Regret Analysis

Regret = “Inner-Regret” + “Outer-Regret”

i) Inner-Regret = $\underbrace{(T/B)}_{\text{\# Epochs}} \cdot \underbrace{\sqrt{B}}_{\text{Regret per epoch}}$

ii) Outer-Regret: $\underbrace{\varepsilon T}_{\text{evict epoch}} + \underbrace{Bn \log n}_{\text{stay epoch}}$

Summing and choosing B properly => εT regret

Online Width Reduction

Second part of our algorithm

- Baseline: $1/\varepsilon^2$ memory and εT regret
- Width reduction: boost the regret guarantee
  - **Observation**: If the loss is within $[0,\rho]$ instead of $[0,1]$, the regret guarantee is $O(\varepsilon\rho T)$ instead of $O(\varepsilon T)$
  - **Observation**: Average loss of any expert is $\geq$ baseline - ε, by the regret guarantee
- Idea: Create meta-expert e_i by taking the best of { *i* and baseline }

Online Width Reduction

- Idea: Create meta-expert e_i by taking the best of { *i* and baseline }
  - Run MWU on *i* and baseline (intuitively, MWU is taking min, up to small gap)
- Claim: the loss of e_i is in [Baseline - ε, Baseline]
  - Upper bound: MWU is ~ taking min
  - Lower bound due to [**Observation**: Average loss of any expert is $\geq$ baseline - ε, by the regret guarantee]
- Reduces width from 1 to ε

Conclusion

Open directions

- Applications of expert learning (treating our algorithm as a blackbox)
  - Streaming (e.g., maximum matching, linear programming and set cover)
  - Finding game equilibria (adaptivity is an issue)
- Extend our results to
  - other sequential decision making problems (e.g., reinforcement learning)
  - other notions of regret (e.g., swap regret and dynamic regret)
- Calibration in online prediction
- Simpler optimal algorithm (c.f. Peng & Rubinstein, arXiv 2023)