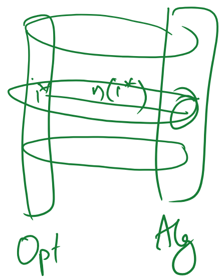
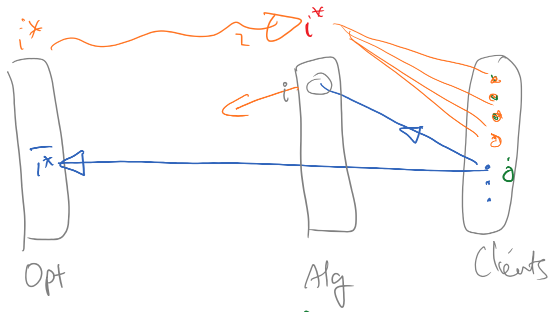


$i^* \longrightarrow i$

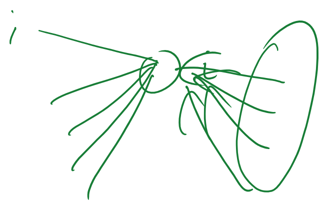
$\bar{i}^* = i^*$

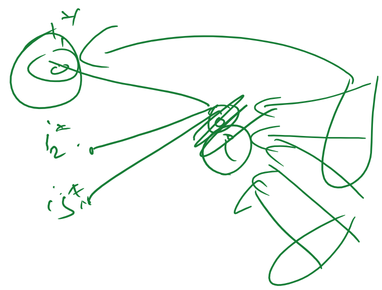


what is $\eta(\bar{i}^*)$?

Send j to $\eta(\bar{i}^*)$ is fine as long as $i \neq \eta(\bar{i}^*)$

If i has degree 1 in η & $\eta(\bar{i}^*) = i$ then $i^* = \bar{i}^*$





Online Algorithms

Paging (CPU cache)

There are 'n' pages $\{1\ 2\ 3\ .\ .\ n\}$

There are 'k' slots

There is a sequence of page requests. (appears online)

Q: What should my eviction policy be, to minimize # evictions

Q: What should my eviction policy ... #evictions

Example

Suppose $n = 3$, (i.e.) 3 pages A, B, C

$k = 2$, 2 cache slots.

Toy example request sequence:–

A A B C A A B B B C A C C

$\boxed{T = 13}$ length of request sequence.

Initial Cache $\boxed{A \mid B}$

$\boxed{C \mid B}$ $\boxed{C \mid A}$ $\boxed{B \mid A}$ $\boxed{B \mid C}$ $\boxed{A \mid C}$

A A A B C B C, B C B C B C . . .

$\boxed{A \mid B} \rightarrow \boxed{A \mid C} \rightarrow \boxed{B \mid C} \rightarrow \boxed{A \mid C}$

OPT = 3 evictions

Q1: if the request sequence is known ahead of time?

{ what is the optimal algorithm? Evict Farthest into Future }

Candidate algorithms for real life "Online" problem

- FIFO
- LRU
- FILO
- Random page evict

COMPETITIVE RATIO OF AN ALGORITHM A

$$CR(A) = \underset{\sigma}{\text{WORST CASE}} \frac{\text{Cost}(Alg_\sigma)}{\text{Opt}(\sigma)} = \frac{\#\,\text{evictions}(A)}{\text{Opt}\cdot\#\,\text{evictions}(\sigma)}$$

competitive Ratio of LRU:–

$$CR(A) = \max \frac{\text{Cost}(A, \sigma)}{}$$

Q: What is CR of ~~FIF~~ LIFO ?

Q: What about LRU ?

Cache size k

pages is $n = k+1$

1 2 3 .. k k+1 1 2 3 .. k k+1 ... 1 2 3 .. k k+1 ...

if $|\sigma| = L$

Cost(LRU) $\approx L - k$

Opt $\gtrsim \frac{L}{k+1}$

$CR \geq \approx k$

Thm: LRU is k-competitive

(ie) $C.R(LRU) \leq k$

σ_1 σ_2 σ_3

k+1 distinct pages   k+1 distinct pages

OPT has to make ≥ 1 eviction in each of the blocks.

LRU makes $\leq k$ evictions in a block.

[Once a page is brought into a block, it won't get evicted until ~~it~~ there are 'k' new distinct requests after this]

Q: Is there a better than k-competitive deterministic algorithm?

~~Consider~~ Lower Bound on CR of any algorithm

Consider any algorithm A.

Sps $n = k+1$

~~a~~ bad sequence: choose the missing page !!

k-1

| 1 | 2 | 3 | . . | . | k |

k+1

If seq (σ) has length L, # evictions $\geq L - \cancel{k}$.

| 1 | 2 | 3 | . . . | k |

$k+1$

If seq (σ) has length L,
#evictions $\geq L - k$.

Opt # evictions (σ) $\leq \frac{L}{k}$

$\uparrow k$

$\boxed{CR \geq k}$

Q: What about randomized algorithms ??

$$CR(A) = \max_{\sigma} \frac{E[\text{Cost}(A, \sigma)]}{\text{Opt}(\sigma)}$$

Note
1) Adversary is "oblivious" to algorithms randomness

Beautiful Result :-
"Randomized LRU" ("Marking" Algorithm)
$= \Theta(\log k)$ - competitive.

$n = 100$ $k = 2$

1 2 3 4 5 6 7 8 . . 100 1 2 3 4 5 . 100

2n

OPT ≥ 1

Online Learning

Given 'N' experts (predicting whether it is going to rain or not)
↓
Using them, we need to make a prediction.
↓
Nature reveals the right answer
↓
go to next day.

Goal: minimize # mistakes we make after T days, when compared to the best expert in hindsight.

Simple Case

There exists a perfect expert who makes no mistakes.
→ There is a simple algorithm, which for any T

makes $\leq \log_2 N$ mistakes

---

① Active set of experts = all experts initially
② On any day, predict what the majority of your active set predicts
③ Once you know the result for the day, delete all experts who made a mistake from the active set.

Proof

① Whenever we make a mistake, size of active set ~~drops~~ by half.

② We'll never throw out the perfect expert

General Case

Suppose there is NO perfect expert. What's a good algorithm?

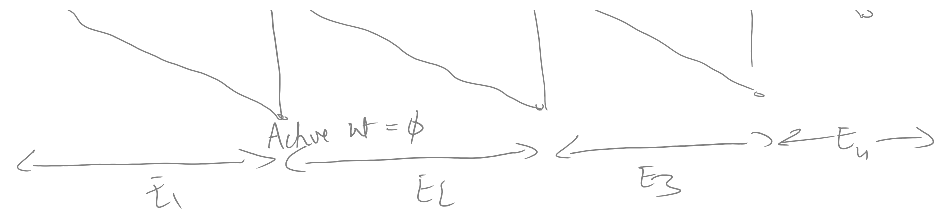
$\boxed{\begin{array}{l}\text{Thm: If the best expert (in hindsight) makes `m' mistakes,} \\ \exists \text{ algo. which makes } \leq m(\log_2 N + 1) + \log_2 N \\ \text{mistakes}\end{array}}$

→ Initialize all experts as active
→ Predict majority of active experts
→ when answer comes, delete all experts who made a mistake from active set
→ if active set $= \phi$, go back to step ①

Thm: $\forall T$, If we run algo for T steps, & expert i has made m_i mistakes in the T steps,

$$\# \text{mistakes we make} \leq m_i(1 + \log_2 N) + \log_2 N$$

Proof: Divide T into epochs



(A) In each epoch, we make $\le \log_2 N + 1$ mistakes.

(B) In each of the epochs (except the last) every expert makes ≥ 1 mistake.

$\Rightarrow$ Overall #mistakes(Alg) $\le m_i (\log_2 N + 1) + \log_2 N$

$\boxed{Q}$ Can we do better than $(\log_2 N + 1)$ - approximation?

$\downarrow$

<u>Idea</u>!! don't be hasty in throwing out experts if they make mistakes.

Assign a <u>weight</u> / confidence for each expert

Initially, all $w_i^{(0)} = 1$ is the confidence

At any time t, go with the prediction with higher total weight.

When answer is revealed,

set $w_i^{(t+1)} = w_i^{(t)} \cdot \frac{1}{2}$ for all incorrect experts i

$= w_i^{(t)}$ for all correct experts i

<u>Theorem</u> :-

For all T, if experts i makes m_i mistakes in T steps, then our algo makes

$\le 3(m_i + \log_2 N)$ mistakes.

<u>Proof</u>

$$\phi^{(t)} = \sum_{i=1}^{N} w_i^{(t)} \qquad \phi^{(0)} = N$$

Note: $\phi^{(t+1)} \le \phi^{(t)}$

<u>If we make a mistake at time t,</u>

If we make a mistake at time t,

$$\phi^{(t+1)} = \sum_{\substack{i \text{ wrong} \\ \text{at time } t}} w_i^{(t+1)} + \sum_{\substack{i \text{ correct} \\ \text{at time } t}} w_i^{(t+1)}$$

$$= \sum_{\substack{i \text{ wrong} \\ \text{at time } t}} \frac{w_i^{(t)}}{2} + \sum_{\substack{i \text{ correct} \\ \text{at time } t}} w_i^{(t)}$$

$$= \phi^{(t)} - \sum_{\substack{i \text{ wrong} \\ \text{at time } t}} \frac{w_i^{(t)}}{2}$$

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

$$\sum_{\substack{i \text{ wrong} \\ \text{at time } t}} w_i^{(t)} \geq \sum_{i \text{ correct}} w_i^{(t)}$$

$$\Rightarrow \sum_{\substack{i \text{ wrong} \\ \text{at time } t}} w_i^{(t)} \geq \frac{1}{2} \phi^{(t)}$$

At time T, if expert i has made m_i mistakes,

$$w_i^{(T)} = \left(\frac{1}{2}\right)^{m_i}$$

$$\Rightarrow \phi^{(T)} \geq \left(\frac{1}{2}\right)^{m_i}$$

At time T, if we made M mistakes by then,

$$\phi^{(T)} \leq \phi^{(0)} \cdot \left(\frac{3}{4}\right)^M$$

$$\Rightarrow \left(\frac{1}{2}\right)^{m_i} \leq \phi^{(T)} \leq \phi^{(0)} \left(\frac{3}{4}\right)^M = N \cdot \left(\frac{3}{4}\right)^M$$

$$\left(\frac{4}{3}\right)^M \leq N \cdot 2^{m_i}$$

$$M \cdot \log_2 \frac{4}{3} \leq \log N + m_i$$

$$M \leq \frac{\log N + m_i}{\log_2 \frac{4}{3}}$$

$$\approx \frac{\log N + m_i}{0.415}$$

$$\leq 3(\log N + m_i)$$

Next qn
why $\frac{1}{2}$ the weights?
try reducing weights of incorrect experts by $(1-\varepsilon)$ for small $\varepsilon > 0$.

If we make mistake,

$$\phi^{(t+1)} \leq \phi^{(t)}\left(1-\frac{\varepsilon}{2}\right)$$

$$\phi^{(T)} \leq \phi^{(0)} \cdot \left(1-\frac{\varepsilon}{2}\right)^{M}$$ if we made M mistakes @ end

if expert i made m_i mistakes

$$\phi^{(T)} \geq w_i^{(T)} \geq (1-\varepsilon)^{m_i}$$

$$(1-\varepsilon)^{m_i} \leq \phi^{(T)} \leq \phi^{(0)} \cdot \left(1-\frac{\varepsilon}{2}\right)^{M} = N\left(1-\frac{\varepsilon}{2}\right)^{M}$$

$$\left(\frac{1}{1-\frac{\varepsilon}{2}}\right)^{M} \leq N\left(\frac{1}{1-\varepsilon}\right)^{m_i}$$

$$M \log\left(\frac{1}{1-\frac{\varepsilon}{2}}\right) \leq \log N + m_i \log\left(\frac{1}{1-\varepsilon}\right)$$

$\frac{1}{1-\frac{\varepsilon}{2}} \approx 1+\frac{\varepsilon}{2}$, $\quad \frac{1}{1-\varepsilon} \approx 1+\varepsilon$

$$\approx \quad M \cdot \frac{\varepsilon}{2} \leq \log N + m_i \cdot \varepsilon$$

$$\boxed{M \leq \frac{2\log N}{\varepsilon} + 2m_i}$$

For deterministic algorithms, $2m_i$ is tight !!

E_1	E_2	Alg		
R	NR	P_1 = R	NR	
R	NR	P_2 = NR	R	
⋮	NR	P_3 = R	NR	
⋮	⋮	⋮	⋮	
R	NR	P_T		

We make mistake every day (total $= T$)

~~Afa~~ At least 1 expert $\leq \frac{T}{2}$ mistakes.

What about randomized algorithms ??

Idea!! Make prediction in proportion to weight!!

$$\boxed{E[\text{Mistakes we make}] \leq (1+\varepsilon) m_i + O\left(\frac{\log N}{\varepsilon}\right)}$$

$\downarrow$

Over time, we're learning who the best expert is !!

$\boxed{\text{Multiplicative Weights Algorithm}}$