Online Algorithms

Paging (CPU Cache)

There are \( n \) pages \( \{1, 2, 3, \ldots, n\} \)

There are \( k \) slots

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

Q: What should my eviction policy be to minimize the number of evictions?
What should my eviction policy be?

Example

Suppose \( n = 3 \), \((c_e) = 3\) pages \( A, B, C \), \( k = 2 \), 2 cache slots.

Toy example request sequence:

<table>
<thead>
<tr>
<th></th>
<th>( A )</th>
<th>( A )</th>
<th>( B )</th>
<th>( A )</th>
<th>( B )</th>
<th>( B )</th>
<th>( C )</th>
<th>( A )</th>
<th>( C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>C</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>B</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Initial Cache

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
</table>

Length of request sequence: \( T = 13 \)

Optimal: \( 3 \) 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:

- \( D \) FIFO
- \( D \) LRU
- \( D \) FILO
- Random page eviction

Competitive Ratio of an Algorithm \( A \):

\[
CR(A) = \frac{\text{Worst Case Cost (Alg) \( (A) \)}}{\text{Opt (\( \sigma \))}} = \frac{\text{Cost (Alg) \( (A) \)}}{\text{Opt (\( \sigma \))}}
\]

Competitive Ratio of LRU:

\[
CR(A) = \max \text{ Cost (A, \( \sigma \))}
\]
Q: What is $CR$ of $LIFO$?

A: What about $LRU$?

Consider $R^k$:

$\#pages = \Theta = k+1$

$1, 2, 3, \ldots, k, k+1, 1, 2, 3, \ldots, k, k+1, \ldots$

If $\text{seq}(R^k) = L$

$\text{Opt}(LRU) \approx \frac{L - k}{k+1}$

$CR \approx \frac{k}{k+1}$

Then $LRU$ is $k$-competitive.

(i.e.) $\text{CR}(LRU) \leq k$

$O_1 \bigg| O_2 \bigg| O_3$

$k+1$ distinct pages

$k+1$ distinct pages

Opt has to make $\geq 1$ 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 $\geq k$ new distinct requests after this.

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

Consider

$	ext{Lower Bound on } CR \text{ of any algorithm}$

Consider any algorithm $A$.

Fix $n = k+1$

With bad sequence $k-1$

Choose the missing page $!!$

If $\text{seq}(A)$ has length $L$,

$\#evictions \geq \frac{L}{k+1}$
Q: What about randomized algorithms?

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

Note

- Adversary is "oblivious" to algorithmic randomness

Beautiful Result:

"Randomized LVU" ("Marking" Algorithm)

\[ = \Theta(\log k) \] competitive.

\[ n = 100 \quad k = 2 \]

[Diagram of values]

Online Learning

Given \( N \) experts (predicting whether it is going to rain or not)

\[ \downarrow \]

Using them, we need to make a prediction.

\[ \downarrow \]

Nature reveals the right answer.

\[ \downarrow \]

Good: 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.

\[ \rightarrow \text{There is a simple algorithm which for any} T \]
makes ≤ \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 agents who made a mistake from the active set.

Proof:
- Whenever we make a mistake, size of active set drops by half.
- We’ll never throw out the perfect expert.

General Case
- Suppose there is NO perfect expert. What’s a good algorithm?

Then:
- If the best expert (in hindsight) makes \( m \) mistakes,
- \( \exists \) algo which makes \( m \left( \log_2 N + 1 \right) + \log_2 N \) mistakes

- Initialize all experts as active
- Predict majority of active experts
- When answer comes, delete all agents who made a mistake from active set
- If active set = \( \emptyset \), go back to step 1.

Then:
- If we ran algo for \( T \) steps, \& expert \( i \) has made \( m_i \) mistakes in the \( T \) steps,
- \# mistakes we make ≤ \( m_i \left( 1 + \log_2 N \right) + \log_2 N \)

Proof: Divide \( T \) into epochs
- all active
In each epoch, we make \( \leq \log_2 N + 1 \) mistakes.

In each of the epochs (except the last) every expert makes \( \geq 1 \) mistake.

\[ \Rightarrow \text{overall mistakes} \leq m_i (\log N + 1) + \log N \]

Can we do better than \( (\log N + 1) \) approximation?

Idea!! don't be lazy in throwing out experts if they make mistakes.

Assign a weight/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

Theorem:
For all \( T \), if experts \( i \) makes \( m_i \) mistakes in \( T \) steps, then our algo makes

\[ \leq 3 (m_i + \log N) \] mistakes.

Proof:

\[ \phi^{(t)} = \sum_{i=1}^{N} w_i^{(t)} \quad \left| \phi^{(0)} = N \right. \]

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

If we make a mistake at time \( t \),

\[ \phi^{(t+1)} \leq \phi^{(t)} \leq \ldots \leq \phi^{(1)} \leq \phi^{(0)} = N \]
If we make a mistake at time $t$,

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

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

$$
= \phi^{(t)} - \sum_{i \text{ wrong at time } t} w_{i}^{(t)}
$$

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

$$
\Rightarrow \sum_{i \text{ wrong at time } t} w_{i}^{(t)} \geq \frac{1}{3}. \phi^{(t)}
$$
Next we ask: why \( \frac{1}{2} \) the weights? try reducing weights if incorrect exparb by \((1-\frac{\epsilon}{2})\) for small \( \epsilon > 0 \).

If we make mistake,
\[
\phi(d^{+}) \leq \phi(d) \left(1 - \frac{\epsilon}{2}\right)
\]

\[
\phi(d) \leq \phi^{(0)} \left(1 - \frac{\epsilon}{2}\right)^{M}
\]

If expert \( i \) made \( m_i \) mistakes
\[
\phi(d^{+}) \leq w_{i}^{(M)} \left(1 - \frac{\epsilon}{2}\right)^{M} = N \left(1 - \frac{\epsilon}{2}\right)^{m_i}
\]

\[
\left(1 - \frac{\epsilon}{2}\right)^{M} \leq N \left( \frac{1}{1 - \frac{\epsilon}{2}} \right)^{m_i}
\]

\[
M \log\left( \frac{1}{1 - \frac{\epsilon}{2}} \right) \leq \log N + m_i \log \left( \frac{1}{1 - \frac{\epsilon}{2}} \right) \approx 1 + \frac{\epsilon}{2}
\]

\[
\approx 1 + \frac{\epsilon}{2}
\]

\[
\frac{M}{2} \leq \log N + m_i \cdot \frac{\epsilon}{2}
\]

\[
M \leq \frac{2 \log N + 2m_i}{\epsilon}
\]

For deterministic algorithm, \( \frac{2m_i}{\epsilon} \) is tight!!
We make mistakes every day \((\text{total } = T)\).

At least 1 expert \(\leq \frac{T}{2}\) mistakes.

What about randomized algorithms? ?

Idea!! Make prediction in proportion to weight!!

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

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

**Multiplicative Weights Algorithm**