Competitive Caching with Machine Learned Advice

Seminar on Online Algorithms

Gal Wiernik

Tel Aviv University

May 25, 2022

Motivation - Machine Learning

Motivation - Machine Learning

- In recent years machine learning algorithms have been **wildly successful**.

Motivation - Machine Learning

- In recent years machine learning algorithms have been **wildly successful**.

Motivation - Machine Learning

- In recent years machine learning algorithms have been **wildly successful**.

- Yet, in pratice:

Motivation - Machine Learning

- In recent years machine learning algorithms have been **wildly successful**.

- Yet, in pratice:
  - Are very difficult to deploy

Motivation - Machine Learning

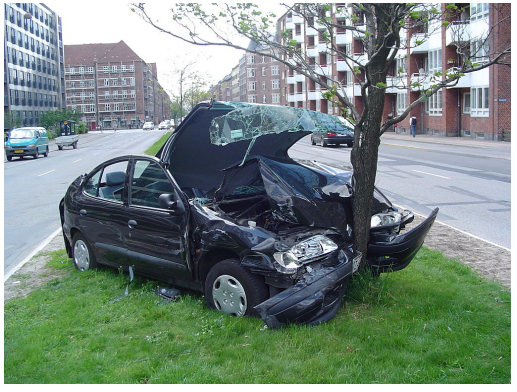
- In recent years machine learning algorithms have been **wildly successful**.

- Yet, in pratice:
  - Are very difficult to deploy
  - Are prone to errors

Motivation - Machine Learning

- In recent years machine learning algorithms have been **wildly successful**.

- Yet, in pratice:
  - Are very difficult to deploy
  - Are prone to errors



Motivation - Online Algorithms

- Online algorithms act without any knowledge of the future.

Motivation - Online Algorithms

- Online algorithms act without any knowledge of the future.
  - Are robust against any input

Motivation - Online Algorithms

- Online algorithms act without any knowledge of the future.
  - Are robust against any input
  - Have a provable gaurantee on performance

Motivation - Online Algorithms

- Online algorithms act without any knowledge of the future.
  - Are robust against any input
  - Have a provable gaurantee on performance
- Yet, overly cautious

Comparison

ML Algorithms	Online Algorithms
attempt to **predict the unknown**	act without any knowledge
suceptible to **large errors**	**robust againt any input**
exploit **patterns**	overly **cautious**

The main question

What if we could combine the **predictive power** of ML with the **robustness** of online algorithms?

First example

Example: Binary Search

Textbook problem - Sorted array A of size n, and query q.
What is the query cost?

First example

Example: Binary Search

Textbook problem - Sorted array A of size n, and query q.
What is the query cost?

ML Approach

First example

Example: Binary Search

Textbook problem - Sorted array A of size n, and query q.
What is the query cost?

ML Approach

- train a classifier $h(q)$ to predict $t(q)$.

First example

Example: Binary Search

Textbook problem - Sorted array A of size n, and query q.
What is the query cost?

ML Approach

- train a classifier $h(q)$ to predict $t(q)$.
- How can we use such a classifier?

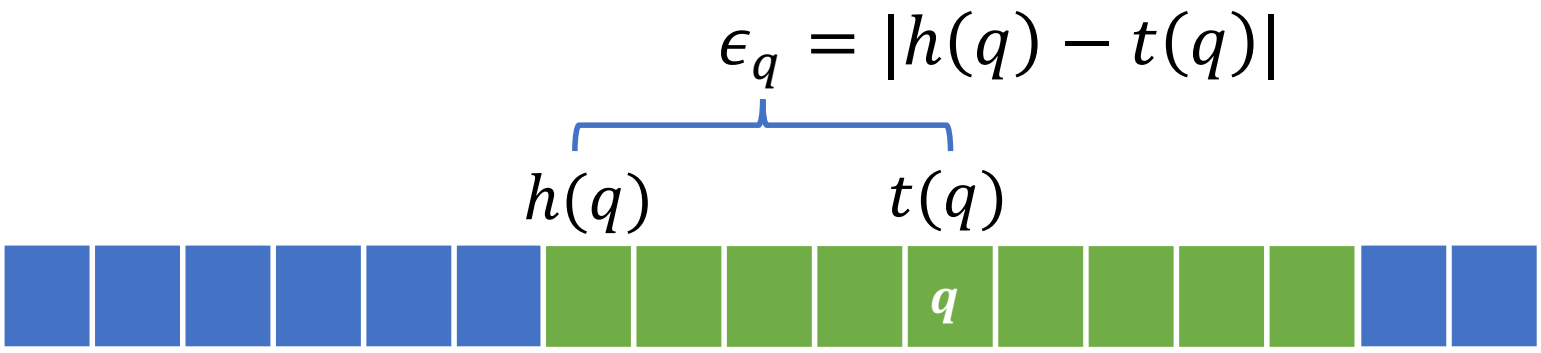

q


$t(q)$
q

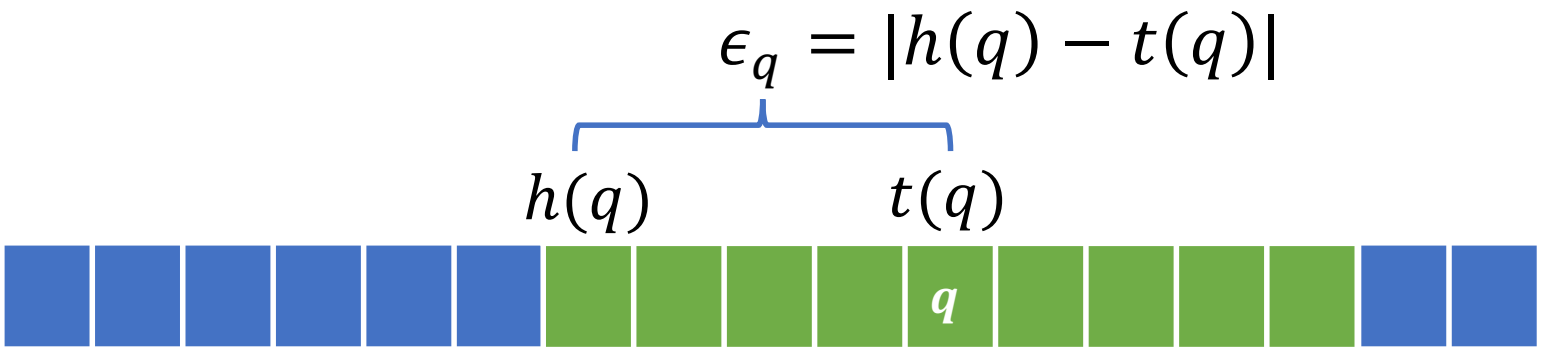

$h(q)$
$t(q)$
q


$h(q)$
$t(q)$
q

$$\epsilon_q = |h(q) - t(q)|$$



The expected cost is $2 \cdot \log(\epsilon_q)$.



The expected cost is $2 \cdot \log(\epsilon_q)$.

Is this any good?

The caching problem

- Our focus will be the **caching problem**.

The caching problem

- Our focus will be the **caching problem**.
- Best **determinstic** algorithm for online caching - $\Theta(k)$ competitive ratio

The caching problem

- Our focus will be the **caching problem**.
- Best **determinstic** algorithm for online caching - $\Theta(k)$ competitive ratio
- Best **randomized** algorithm - $\Theta(\log k)$

The caching problem

- Our focus will be the **caching problem**.
- Best **determinstic** algorithm for online caching - $\Theta(k)$ competitive ratio
- Best **randomized** algorithm - $\Theta(\log k)$
  - In reality, obsereved competitive ratio is much lower.

The caching problem

- Our focus will be the **caching problem**.
- Best **determinstic** algorithm for online caching - $\Theta(k)$ competitive ratio
- Best **randomized** algorithm - $\Theta(\log k)$
  - In reality, obsereved competitive ratio is much lower.

The main result

The **machine-learning assisted algorithm** reaches a competitive ratio of $2 + O\left(\min\left(\sqrt{\epsilon}, \log k\right)\right)$.

Preliminaries

- To achieve our results we have to define the playing ground for a **new genere of algorithms**: competitive algorithms with machine learning advice.

Preliminaries

- **ML scenarios** consist of:
  - Feature space - $\mathcal{X}$, and labels - $\mathcal{Y}$
  - hypothesis $h : \mathcal{X} \to \mathcal{Y}$
  - loss function: $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$

Preliminaries

- **ML scenarios** consist of:
  - Feature space - $\mathcal{X}$, and labels - $\mathcal{Y}$
  - hypothesis $h : \mathcal{X} \to \mathcal{Y}$
  - loss function: $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$
  - The loss can be: abosolute ($\ell_1(y, \hat{y}) = |y - \hat{y}|$), squared ($\ell_2(y, \hat{y}) = (y - \hat{y})^2$), or generally: $\ell_c(y, \hat{y}) = \mathbf{1}_{y \neq \hat{y}}$

Preliminaries

- **ML scenarios** consist of:
  - Feature space - $\mathcal{X}$, and labels - $\mathcal{Y}$
  - hypothesis $h : \mathcal{X} \to \mathcal{Y}$
  - loss function: $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$
  - The loss can be: abosolute ($\ell_1(y, \hat{y}) = |y - \hat{y}|$), squared ($\ell_2(y, \hat{y}) = (y - \hat{y})^2$), or generally: $\ell_c(y, \hat{y}) = \mathbf{1}_{y \neq \hat{y}}$
- **Online scenarios** consist of a algorithm $\mathcal{A}$ and sequences σ.

Preliminaries

- **ML scenarios** consist of:
  - Feature space - $\mathcal{X}$, and labels - $\mathcal{Y}$
  - hypothesis $h : \mathcal{X} \to \mathcal{Y}$
  - loss function: $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$
  - The loss can be: abosolute ($\ell_1(y, \hat{y}) = |y - \hat{y}|$), squared ($\ell_2(y, \hat{y}) = (y - \hat{y})^2$), or generally: $\ell_c(y, \hat{y}) = \mathbf{1}_{y \neq \hat{y}}$
- **Online scenarios** consist of a algorithm $\mathcal{A}$ and sequences σ.
- $\mathcal{A}$ has comptitive ratio CR if for every σ:
  $cost_{\mathcal{A}}(\sigma) \leq \mathrm{CR} \cdot \mathrm{OPT}(\sigma)$

The Online ML-Assisted Framework

- Now we can define the combined framework.

The Online ML-Assisted Framework

- Now we can define the combined framework.
- We have a **universe** $\mathcal{Z}$ and **feature** space $\mathcal{X}$

The Online ML-Assisted Framework

- Now we can define the combined framework.
- We have a **universe** $\mathcal{Z}$ and **feature** space $\mathcal{X}$
- The input is a sequence of items $\sigma = (\sigma_1, \sigma_2, \dots)$
  Each item σ_i is assosiated with an element $z_i \in \mathcal{Z}$ and with features $x_i \in \mathcal{X}$

The Online ML-Assisted Framework

- Each item σ_i also has a label $y_i \in \mathcal{Y}$

The Online ML-Assisted Framework

- Each item σ_i also has a label $y_i \in \mathcal{Y}$
- A predictor $h : \mathcal{X} \to \mathcal{Y}$ returns $h(\sigma_i)$, attempts to find y_i

The Online ML-Assisted Framework

- Each item σ_i also has a label $y_i \in \mathcal{Y}$
- A predictor $h : \mathcal{X} \to \mathcal{Y}$ returns $h(\sigma_i)$, attempts to find y_i
  - The total loss of h on σ is:

$$\eta_\ell(h, \sigma) = \sum_i \ell(h(\sigma_i), y_i)$$

The Online ML-Assisted Framework

- Each item σ_i also has a label $y_i \in \mathcal{Y}$
- A predictor $h : \mathcal{X} \to \mathcal{Y}$ returns $h(\sigma_i)$, attempts to find y_i
  - The total loss of h on σ is:

$$\eta_\ell(h, \sigma) = \sum_i \ell(h(\sigma_i), y_i)$$

Question

How can we define h to have a general accuracy of ϵ?

The Online ML-Assisted Framework

- Each item σ_i also has a label $y_i \in \mathcal{Y}$
- A predictor $h : \mathcal{X} \to \mathcal{Y}$ returns $h(\sigma_i)$, attempts to find y_i
  - The total loss of h on σ is:

$$\eta_\ell(h, \sigma) = \sum_i \ell(h(\sigma_i), y_i)$$

Question

How can we define h to have a general accuracy of ϵ?

Definition

We say that h is $\epsilon - \textit{accurate}$ if for every σ,
$\eta_\ell(h, \sigma) \leq \epsilon \cdot \textsc{Opt}(\sigma)$.

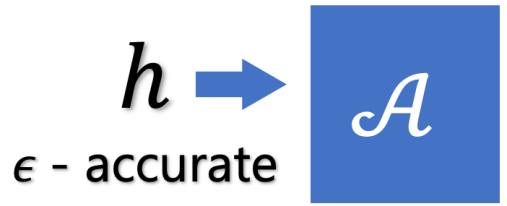
The Online ML-Assisted Framework

The Online ML-Assisted Framework

Define $\mathrm{CR}_{\mathcal{A}}(\epsilon)$ to be the **competitive ratio** of algorithm $\mathcal{A}$ that uses any predictor h that is ϵ-accurate.

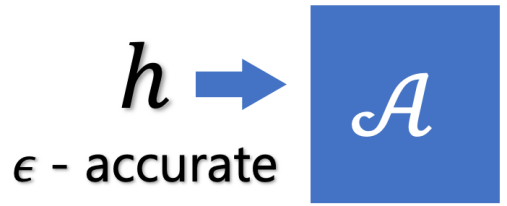
The Online ML-Assisted Framework

Define $\mathrm{CR}_{\mathcal{A}}(\epsilon)$ to be the **competitive ratio** of algorithm $\mathcal{A}$ that uses any predictor h that is ϵ-accurate.



The Online ML-Assisted Framework

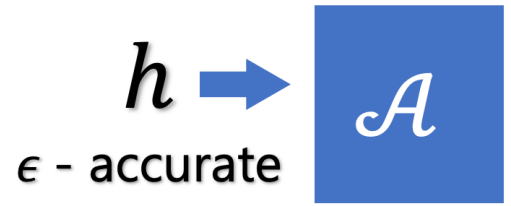
Define $\mathrm{CR}_{\mathcal{A}}(\epsilon)$ to be the **competitive ratio** of algorithm $\mathcal{A}$ that uses any predictor h that is ϵ-accurate.



- What would we like $\mathrm{CR}_{\mathcal{A}}(0)$ to be?

The Online ML-Assisted Framework

Define $\mathrm{CR}_{\mathcal{A}}(\epsilon)$ to be the **competitive ratio** of algorithm $\mathcal{A}$ that uses any predictor h that is ϵ-accurate.

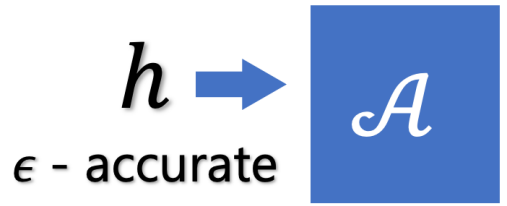


Consistency

$\mathcal{A}$ is β-consistent if $\mathrm{CR}_{\mathcal{A}}(0) = \beta$.

The Online ML-Assisted Framework

Define $\mathrm{CR}_{\mathcal{A}}(\epsilon)$ to be the **competitive ratio** of algorithm $\mathcal{A}$ that uses any predictor h that is ϵ-accurate.



Consistency

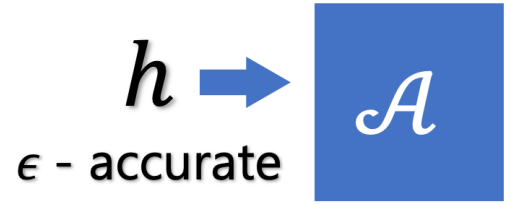
$\mathcal{A}$ is β-consistent if $\mathrm{CR}_{\mathcal{A}}(0) = \beta$.

Robustness

$\mathcal{A}$ is α-robust for some function α if $\mathrm{CR}_{\mathcal{A}}(\epsilon) = O(\alpha(\epsilon))$.

The Online ML-Assisted Framework

Define $\mathrm{CR}_{\mathcal{A}}(\epsilon)$ to be the **competitive ratio** of algorithm $\mathcal{A}$ that uses any predictor h that is ϵ-accurate.



Consistency

$\mathcal{A}$ is β-consistent if $\mathrm{CR}_{\mathcal{A}}(0) = \beta$.

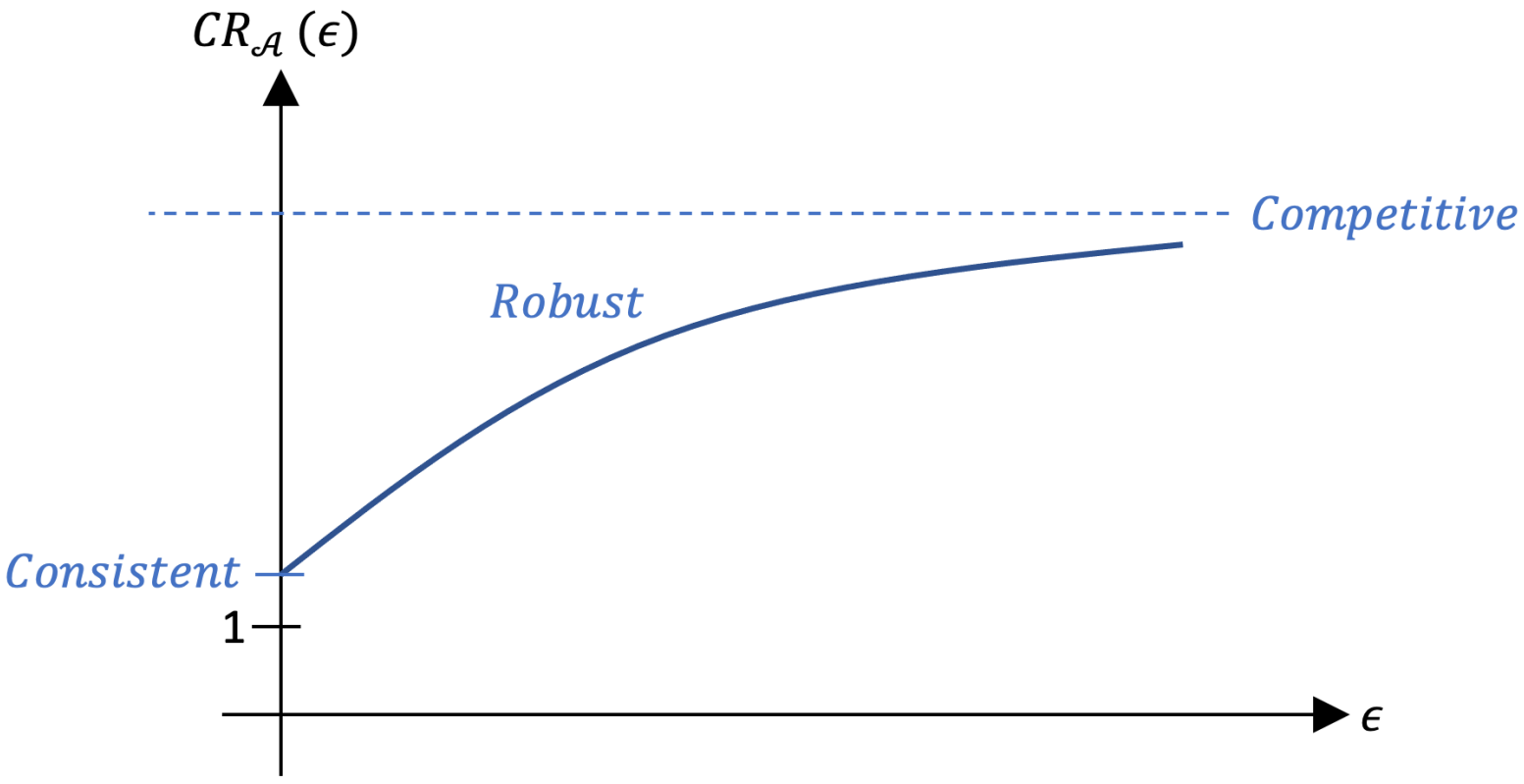
Robustness

$\mathcal{A}$ is α-robust for some function α if $\mathrm{CR}_{\mathcal{A}}(\epsilon) = O(\alpha(\epsilon))$.

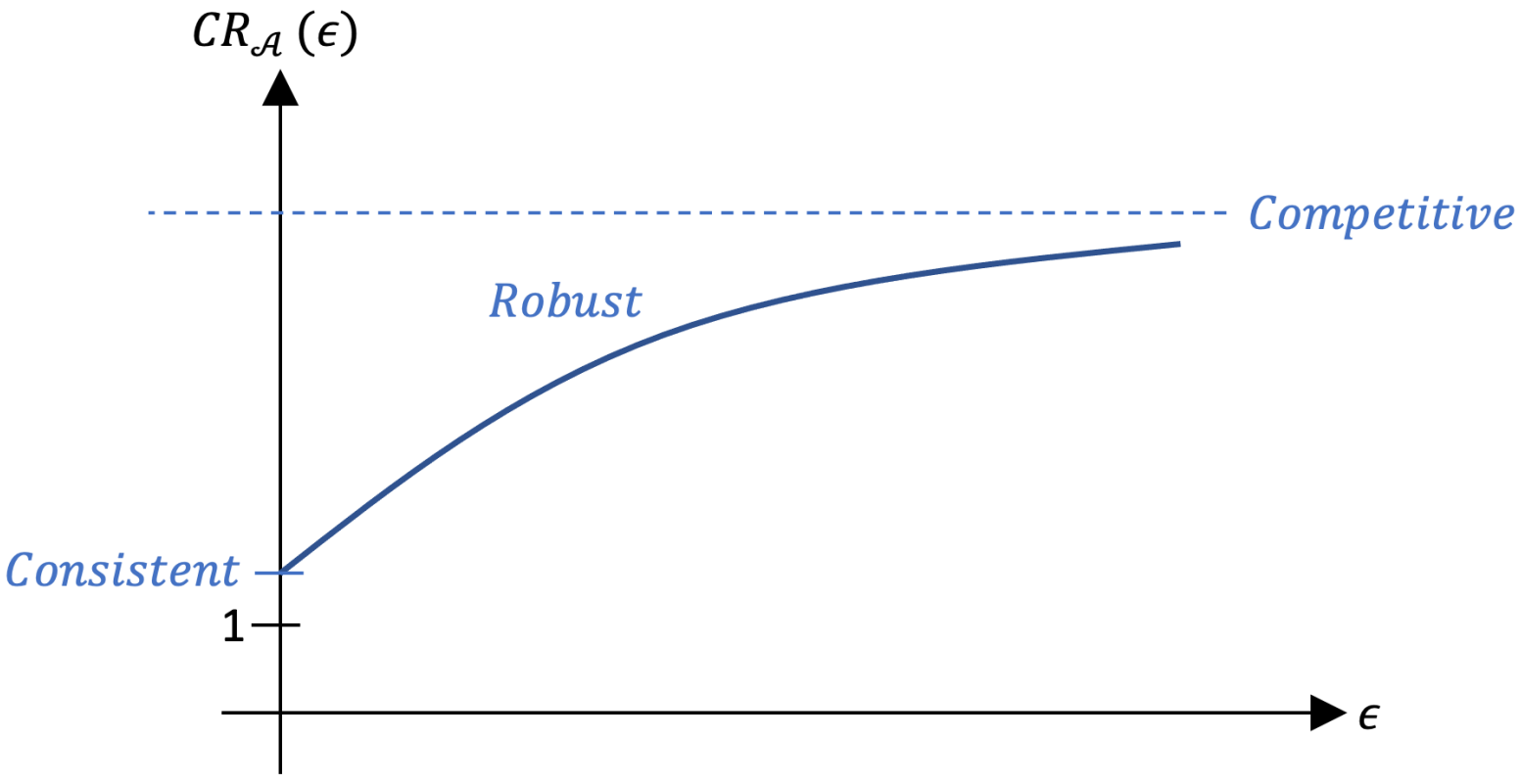
Competitivness

$\mathcal{A}$ is γ-competitive if $\mathrm{CR}_{\mathcal{A}}(\epsilon) \leq \gamma \cdot \mathrm{OPT}$ for all ϵ.

The Online ML-Assisted Framework



The Online ML-Assisted Framework



- The holy grail is an algorithm $\mathcal{A}$ which simultaneously optimizes all three properties.

The Caching Scenario

The Caching Scenario

- Each σ_i is a request

The Caching Scenario

- Each σ_i is a request
- z_i is the requested page, and x_i is features of the request.

The Caching Scenario

- Each σ_i is a request
- z_i is the requested page, and x_i is features of the request.

Question

What should the labels $\mathcal{Y}$ be?
Hint: The optimal caching algorithm is LFD.

The Caching Scenario

- Each σ_i is a request
- z_i is the requested page, and x_i is features of the request.

Question

What should the labels $\mathcal{Y}$ be?
Hint: The optimal caching algorithm is LFD.

- $y(\sigma_i)$ will be the next appearence of z_i.

The Caching Scenario

- Each σ_i is a request
- z_i is the requested page, and x_i is features of the request.

Question

What should the labels $\mathcal{Y}$ be?
Hint: The optimal caching algorithm is LFD.

- $y(\sigma_i)$ will be the next appearence of z_i.
- $h(\sigma_i)$ will try to predict y_i.

The Caching Scenario

- Each σ_i is a request
- z_i is the requested page, and x_i is features of the request.

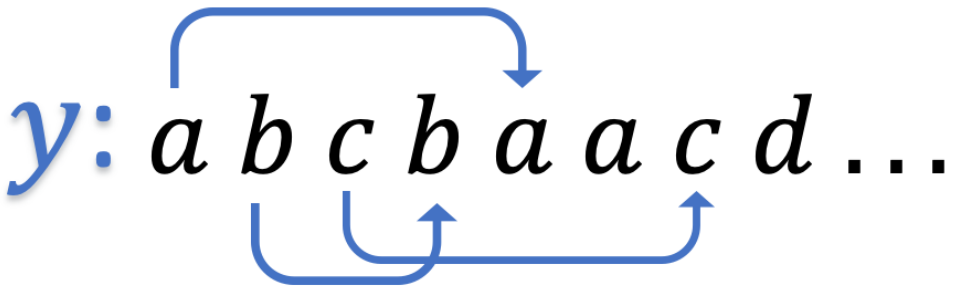
Question

What should the labels $\mathcal{Y}$ be?
Hint: The optimal caching algorithm is LFD.

- $y(\sigma_i)$ will be the next appearence of z_i.
- $h(\sigma_i)$ will try to predict y_i.
- $\mathcal{Y} = \mathbb{N}^+$

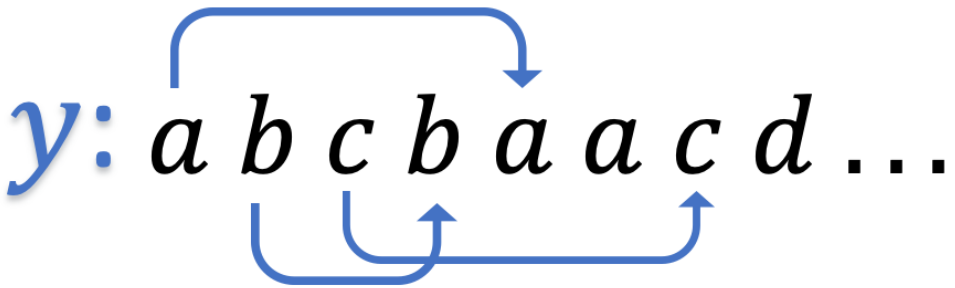
The Caching Scenario

- Example:



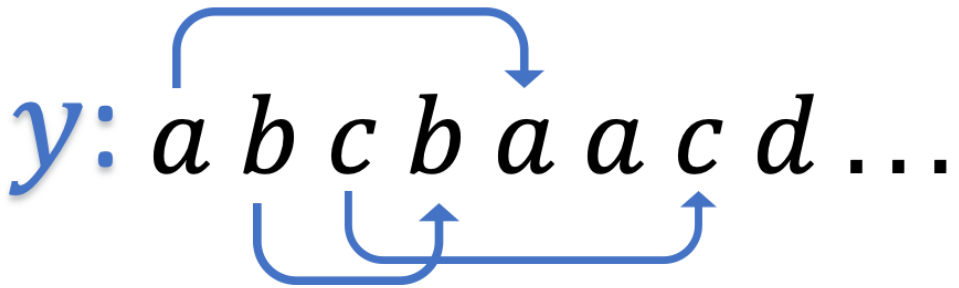
The Caching Scenario

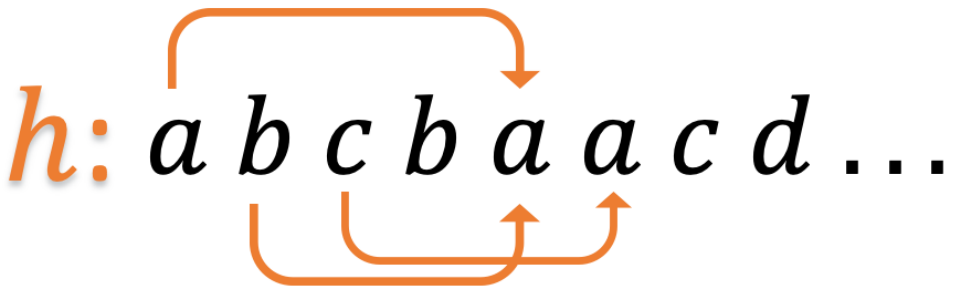
- Example:



The Caching Scenario

- Example:





Attempt #1 - Blindly Following the Predictor

- If the predictor is good, can't we just use it?

Attempt #1 - Blindly Following the Predictor

- If the predictor is good, can't we just use it?

Lemma

Define $\mathcal{B}$ the algorithm that blindly follows an ϵ-accurate predictor. Then $\mathcal{B}$ has a competitive ratio $\mathrm{CR}_{\mathcal{B}}(\epsilon) = \Omega(\epsilon)$.

Attempt #1 - Blindly Following the Predictor

- If the predictor is good, can't we just use it?

Lemma

Define $\mathcal{B}$ the algorithm that blindly follows an ϵ-accurate predictor. Then $\mathcal{B}$ has a competitive ratio $\mathrm{CR}_{\mathcal{B}}(\epsilon) = \Omega(\epsilon)$.

- Assume $k = 2$ and three elements, a, b, c.

Attempt #1 - Blindly Following the Predictor

- If the predictor is good, can't we just use it?

Lemma

Define $\mathcal{B}$ the algorithm that blindly follows an ϵ-accurate predictor. Then $\mathcal{B}$ has a competitive ratio $\mathrm{CR}_{\mathcal{B}}(\epsilon) = \Omega(\epsilon)$.

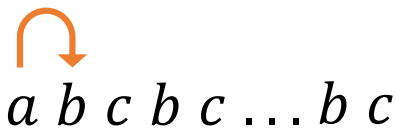
- Assume $k = 2$ and three elements, a, b, c.
- Example follows.

$a\ b\ c\ b\ c\ \ldots\ b\ c$

Predictor h: $true, except\ h(\sigma_1) = 2$

$$a\ b\ c\ b\ c \ldots b\ c$$

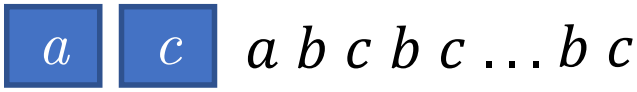
Predictor h: $true, except\ h(\sigma_1) = 2$



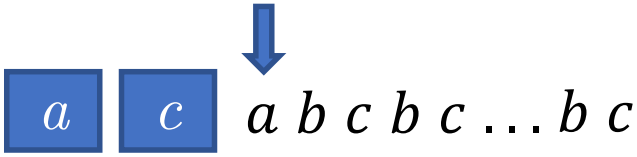
Predictor h: $true, except\ h(\sigma_1) = 2$

$$a\ b\ c\ b\ c \ldots b\ c$$

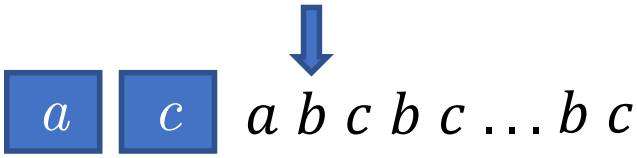
Predictor h: $true, except\ h(\sigma_1) = 2$



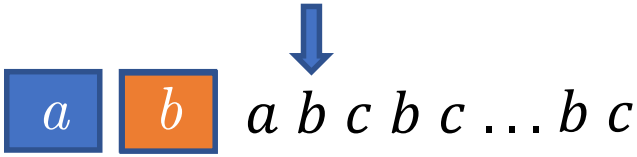
Predictor h: $true, except\ h(\sigma_1) = 2$



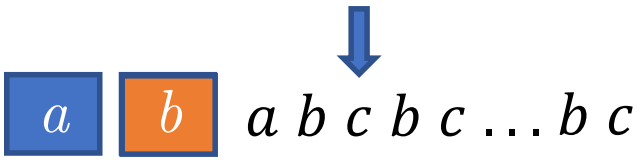
Predictor h: $true, except\ h(\sigma_1) = 2$



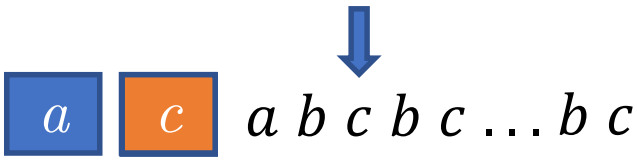
Predictor h: $true, except\ h(\sigma_1) = 2$



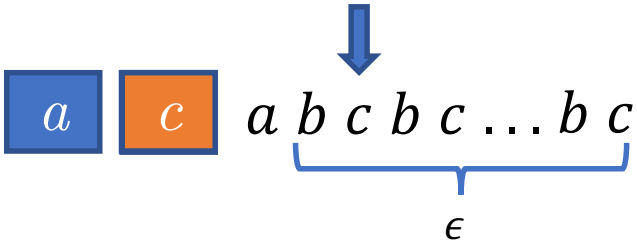
Predictor h: $true, except\ h(\sigma_1) = 2$



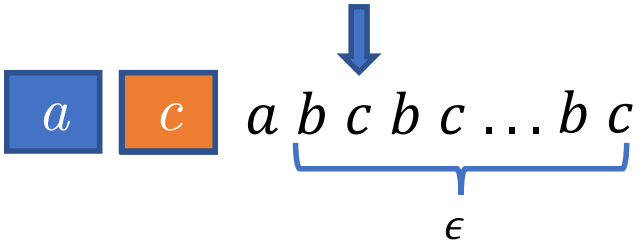
Predictor h: $true, except\ h(\sigma_1) = 2$



Predictor h: $true, except\ h(\sigma_1) = 2$

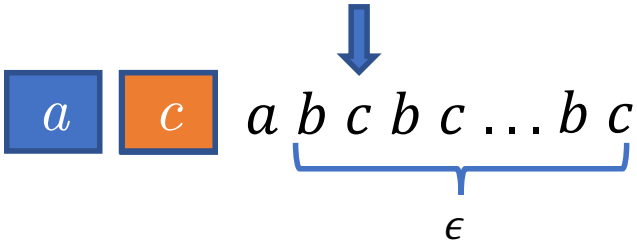


Predictor h: $true, except\ h(\sigma_1) = 2$



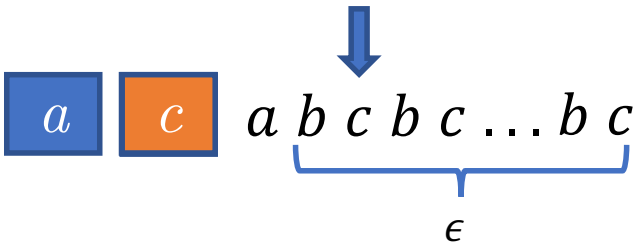
Offline Optimum:

Predictor h: $true, except\ h(\sigma_1) = 2$



Offline Optimum: $OPT = 1$

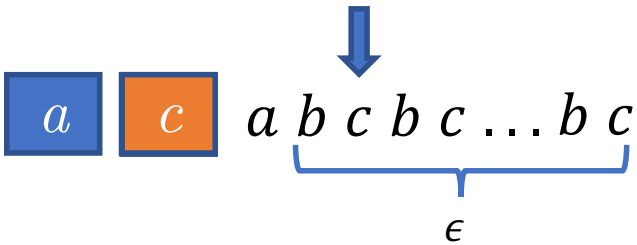
Predictor h: $true, except\ h(\sigma_1) = 2$



Offline Optimum: $OPT = 1$

Performance of $\mathcal{B}$:

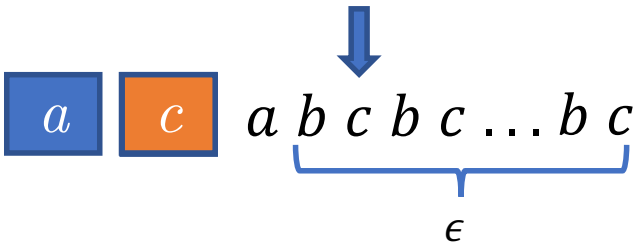
Predictor h: $true, except\ h(\sigma_1) = 2$



Offline Optimum: $OPT = 1$

Performance of $\mathcal{B}$: ϵ

Predictor h: $true, except\ h(\sigma_1) = 2$

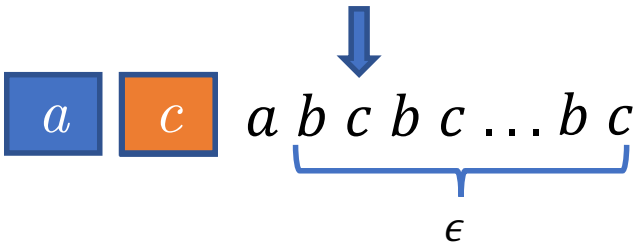


Offline Optimum: $OPT = 1$

Performance of $\mathcal{B}$: ϵ

Absolute Loss: $\eta(h, \sigma) = \varepsilon$

Predictor h: $true, except\ h(\sigma_1) = 2$

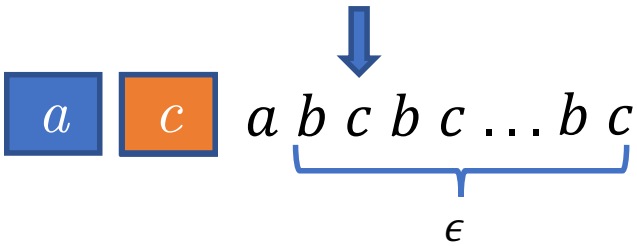


Offline Optimum: $OPT = 1$

Performance of $\mathcal{B}$: ϵ

Absolute Loss: $\eta(h, \sigma) = \varepsilon \leq \epsilon \cdot OPT$

Predictor h: $true, except\ h(\sigma_1) = 2$



Offline Optimum: $OPT = 1$

Performance of $\mathcal{B}$: ϵ

Absolute Loss: $\eta(h, \sigma) = \varepsilon \leq \epsilon \cdot OPT$

Competitive ratio: $CR_{\mathcal{B}}(\epsilon) = \frac{\epsilon}{1} = \Omega(\epsilon)$

Attempt #2 - Reacting to Predictor Mistakes

- We trusted the predictor too much. Can we do better?

Attempt #2 - Reacting to Predictor Mistakes

- We trusted the predictor too much. Can we do better?

Lemma

Define $\mathcal{W}$ the algorithm that follows an ϵ-accurate predictor, but evicts wrong predictions. Then $\mathcal{W}$ has a competitive ratio $\mathrm{CR}_{\mathcal{W}}(\epsilon) = \Omega(\epsilon)$.

Attempt #2 - Reacting to Predictor Mistakes

- We trusted the predictor too much. Can we do better?

Lemma

Define $\mathcal{W}$ the algorithm that follows an ϵ-accurate predictor, but evicts wrong predictions. Then $\mathcal{W}$ has a competitive ratio $\mathrm{CR}_{\mathcal{W}}(\epsilon) = \Omega(\epsilon)$.

- Assume $k = 3$ and four elements, a, b, c, d.

Attempt #2 - Reacting to Predictor Mistakes

- We trusted the predictor too much. Can we do better?

Lemma

Define $\mathcal{W}$ the algorithm that follows an ϵ-accurate predictor, but evicts wrong predictions. Then $\mathcal{W}$ has a competitive ratio $\mathrm{CR}_{\mathcal{W}}(\epsilon) = \Omega(\epsilon)$.

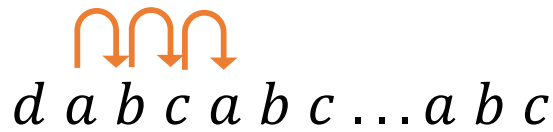
- Assume $k = 3$ and four elements, a, b, c, d.
- Example follows.

$d\ a\ b\ c\ a\ b\ c \ldots a\ b\ c$

Predictor h: $predict\ d\ correctly$
$but\ predict\ a, b, c\ two\ steps\ too\ early$

$$d\ a\ b\ c\ a\ b\ c\ \ldots a\ b\ c$$

Predictor h: $predict\ d\ correctly$
$but\ predict\ a, b, c\ two\ steps\ too\ early$



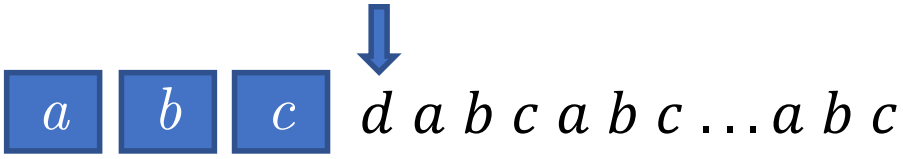
Predictor h: $predict\ d\ correctly$
$but\ predict\ a, b, c\ two\ steps\ too\ early$

$$d\ a\ b\ c\ a\ b\ c\ \ldots a\ b\ c$$

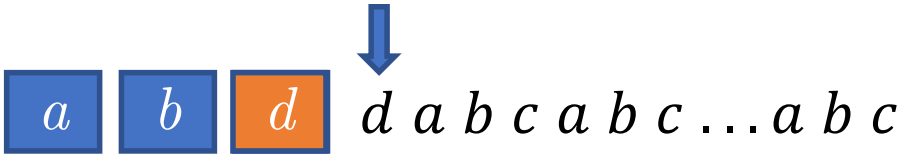
Predictor h: $predict\ d\ correctly$
$but\ predict\ a, b, c\ two\ steps\ too\ early$

$d\ a\ b\ c\ a\ b\ c \ldots a\ b\ c$

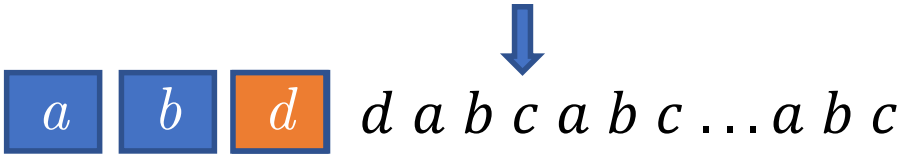
Predictor h: $predict\ d\ correctly$
$but\ predict\ a, b, c\ two\ steps\ too\ early$



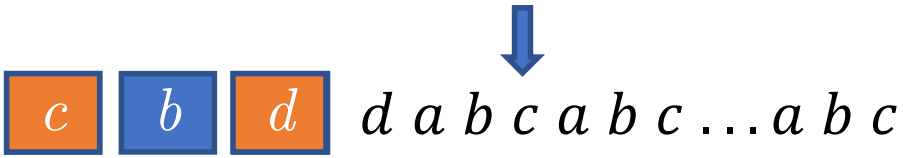
Predictor h: $predict\ d\ correctly$
$but\ predict\ a, b, c\ two\ steps\ too\ early$



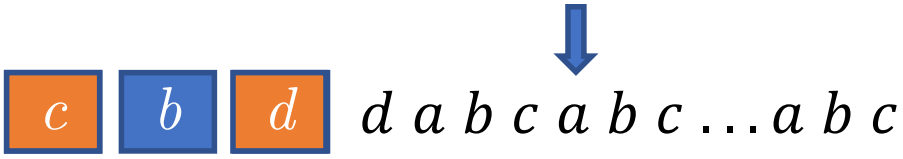
Predictor h: $predict\ d\ correctly$
$but\ predict\ a, b, c\ two\ steps\ too\ early$



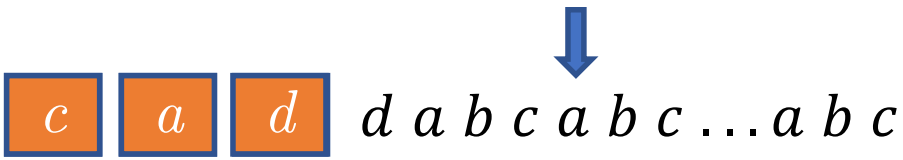
Predictor h: $predict\ d\ correctly$
$but\ predict\ a, b, c\ two\ steps\ too\ early$



Predictor h: $predict\ d\ correctly$
$but\ predict\ a, b, c\ two\ steps\ too\ early$



Predictor h: $predict\ d\ correctly$
$but\ predict\ a, b, c\ two\ steps\ too\ early$



Attempt #3 - Popular Heuristics

- In the examples we saw that there is some element that **should** have been evicted.

Attempt #3 - Popular Heuristics

- In the examples we saw that there is some element that **should** have been evicted.
- LRU and FIFO both provide strong hueristics for such cases.

Attempt #3 - Popular Heuristics

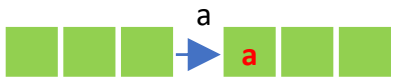
- In the examples we saw that there is some element that **should** have been evicted.
- LRU and FIFO both provide strong hueristics for such cases.
  - However, their **strict** (determisitic) policy leads to weak gaurantees.

Reminder - Classic Marking Algorithm

- Recall the classic Marking Algorithm.


a b c c d c d a b b e a b

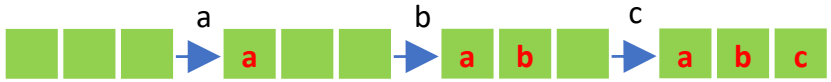

a b c c d c d a b b e a b


a
a


a b c c d c d a b b e a b

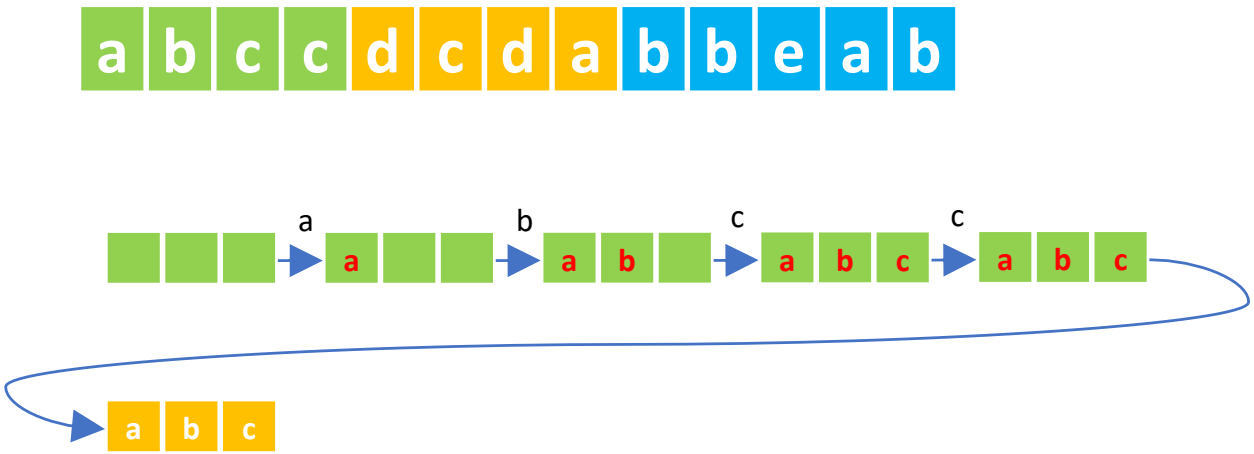

a
b
a
a
b

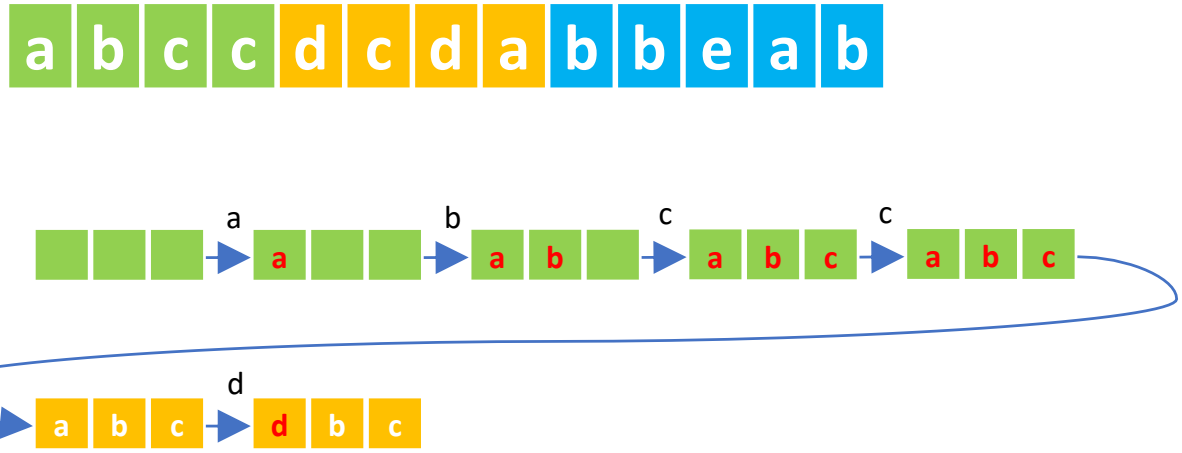

a b c c d c d a b b e a b

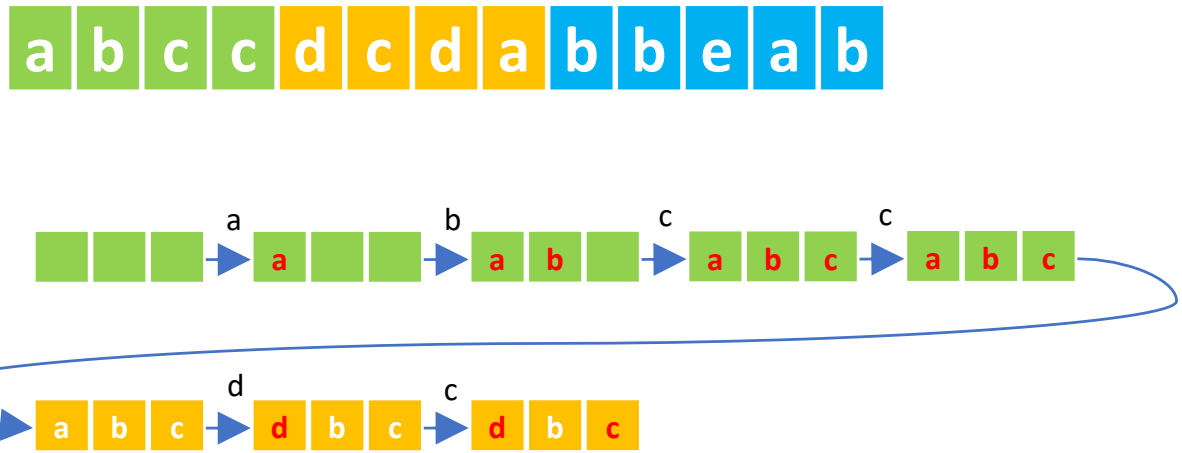

a
b
c
a
a b
a b c

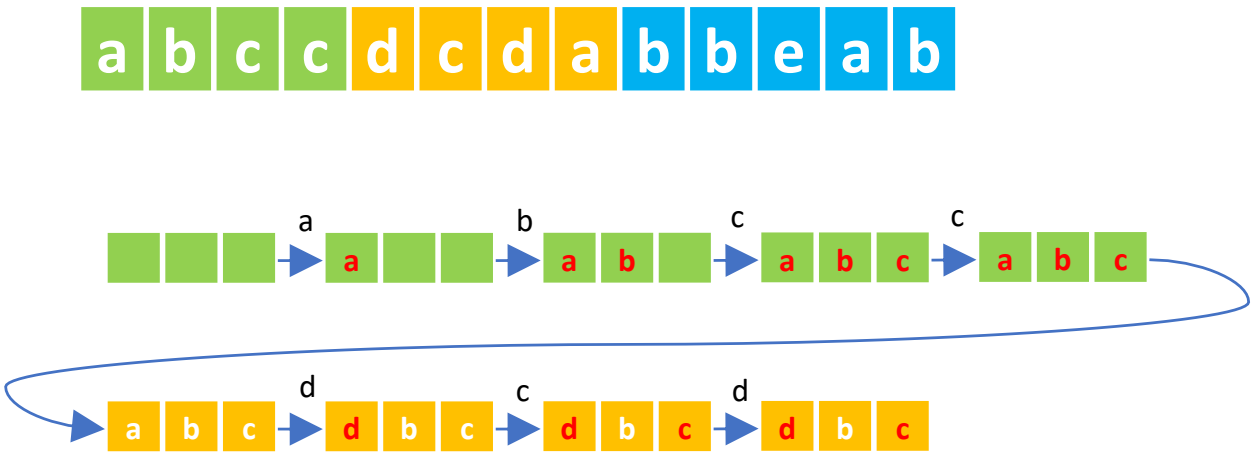

a b c c d c d a b b e a b

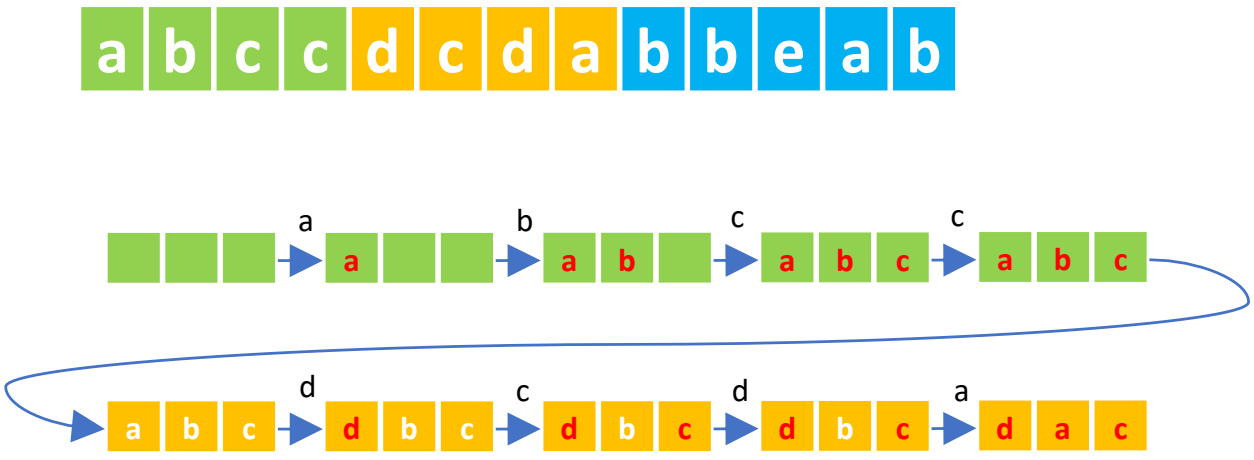

a
b
c
c
a
a b
a b c
a b c

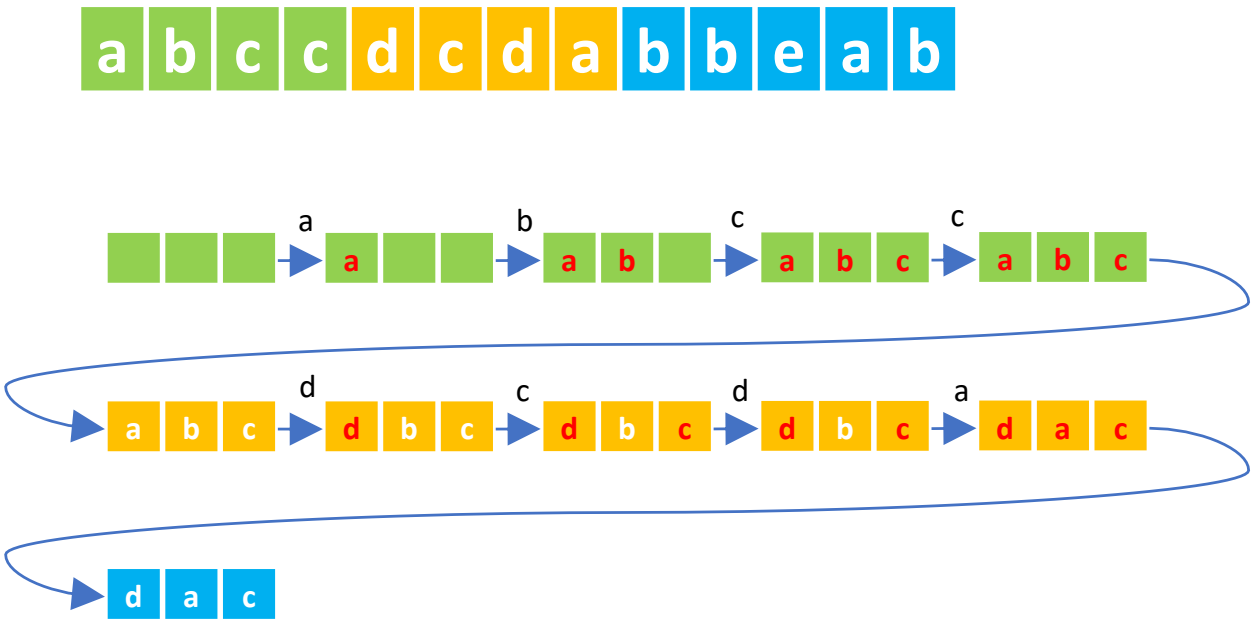

a b c c d c d a b b e a b
a
b
c
c
a b c
a
a b
a b c
a b c
a b c


a b c c d c d a b b e a b
a
b
c
c
d


a b c c d c d a b b e a b
a
b
c
c
d
c


a b c c d c d a b b e a b
a
b
c
c
a b c
d
c
d
a b c d b c d b c d b c


a b c c d c d a b b e a b
a
b
c
c
a b c
d
c
d
a
a b c d b c d b c d b c d a c


a b c c d c d a b b e a b
a
b
c
c
a b c
d
c
d
a
a b c
d b c
d b c
d b c
d a c
d a c

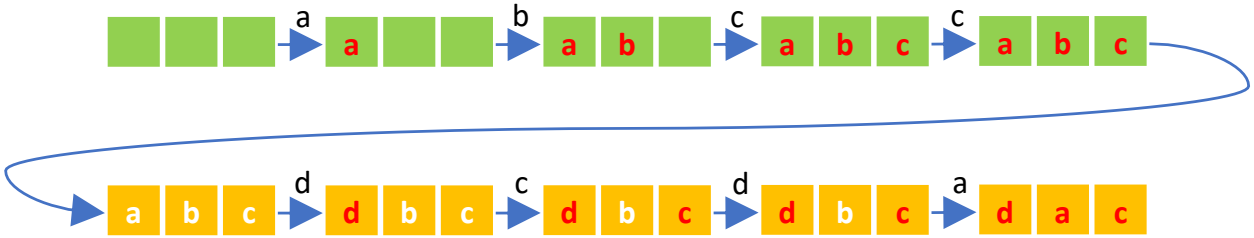
Reminder - Classic Marking Algorithm

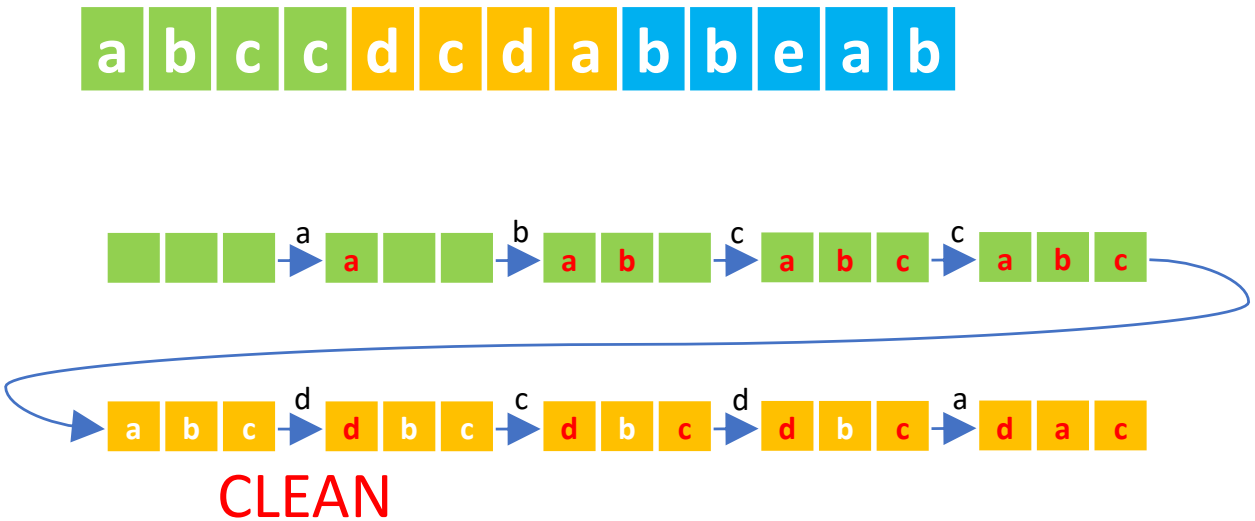
- A **clean** element is an element that didn't arrive in phase $r-1$ and arrives in phase r.

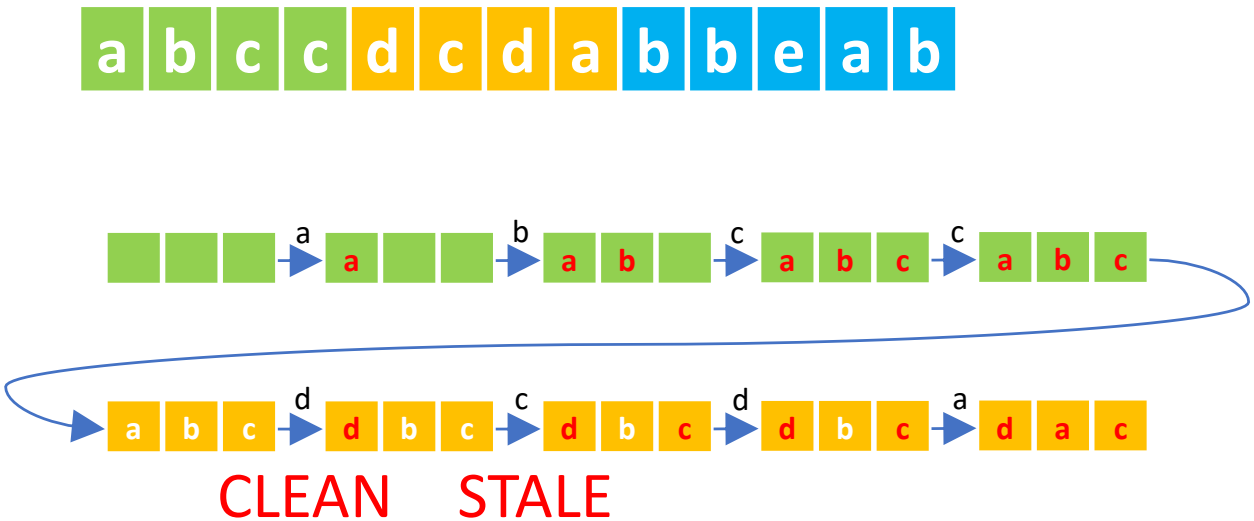
Reminder - Classic Marking Algorithm

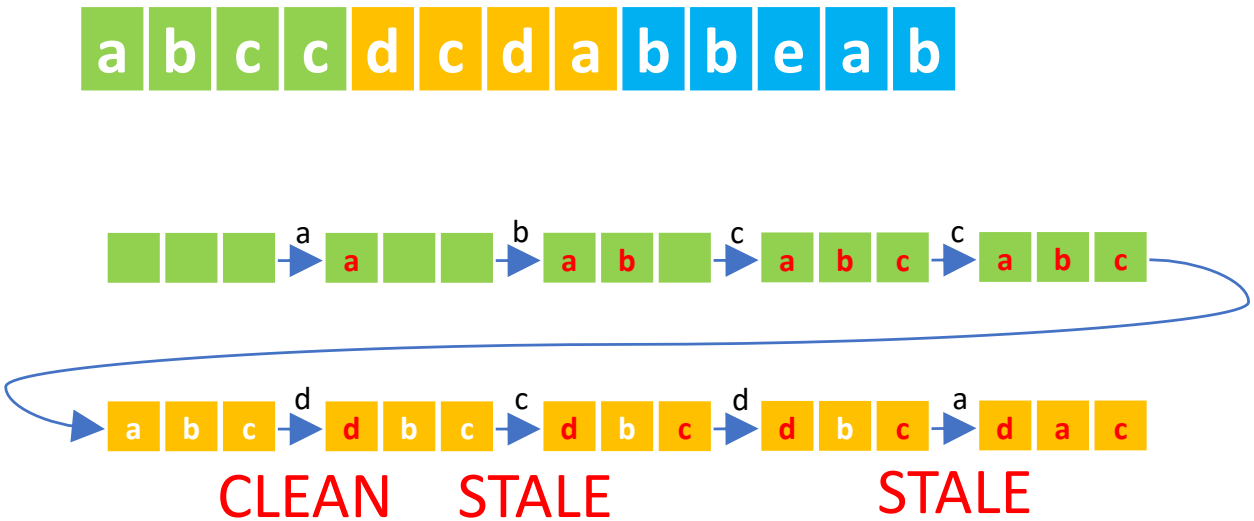
- A **clean** element is an element that didn't arrive in phase $r-1$ and arrives in phase r.
- A **stale** element is an element that arrived in phase $r-1$ and **also** in phase r.


a b c c d c d a b b e a b


a
b
c
c
d
c
d
a
a b c
a b c
a b c
a b c
a b c
d b c
d b c
d b c
d a c


a b c c d c d a b b e a b
a
b
c
c
a b c
d
c
d
a
CLEAN


a b c c d c d a b b e a b
a
b
c
d
c
d
a
a b c d b c c d b c d d b c a d a c
CLEAN
STALE


a b c c d c d a b b e a b
a
b
c
d
c
d
a
CLEAN
STALE
STALE

Reminder - Classic Marking Algorithm

- We saw that the marking algorithm has competitive ratio $O(\log k)$.

Reminder - Classic Marking Algorithm

- We saw that the marking algorithm has competitive ratio $O(\log k)$.
- This came from 2 claims:

Reminder - Classic Marking Algorithm

- We saw that the marking algorithm has competitive ratio $O(\log k)$.
- This came from 2 claims:

Claim 1

Let L be the number of clean elements in σ. Then $\textsc{Opt}(\sigma) \geq \frac{L}{2}$.

Reminder - Classic Marking Algorithm

- We saw that the marking algorithm has competitive ratio $O(\log k)$.
- This came from 2 claims:

Claim 1

Let L be the number of clean elements in σ. Then $\textsc{Opt}(\sigma) \geq \frac{L}{2}$.

Claim 2

Let L be the number of clean elements in σ.
Then $\mathbb{E}[MARK(\sigma)] \leq L \cdot H_k$.

Reminder - Classic Marking Algorithm

- We saw that the marking algorithm has competitive ratio $O(\log k)$.
- This came from 2 claims:

Claim 1

Let L be the number of clean elements in σ. Then $\textsc{Opt}(\sigma) \geq \frac{L}{2}$.

Claim 2

Let L be the number of clean elements in σ.
Then $\mathbb{E}[MARK(\sigma)] \leq L \cdot H_k$.

- Combined, we got:
  $\mathbb{E}[MARK(\sigma)] \leq L \cdot H_k \leq 2 \log k \cdot \textsc{Opt}(\sigma)$.

Predictive Marker

- If we'll use the marking algorithm, we'll gain $O(\log k)$ competitive ratio. How can we improve that?

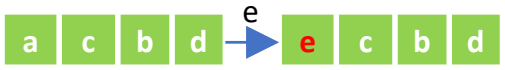
Predictive Marker

- If we'll use the marking algorithm, we'll gain $O(\log k)$ competitive ratio. How can we improve that?
- The natural thing to do then is to **break ties in eviction** using the predictor.

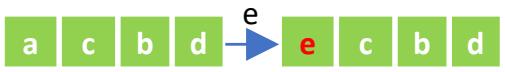
Chains

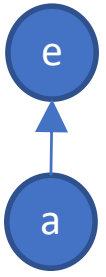
Chains, Explained

Chains, Explained

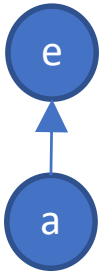


Chains, Explained

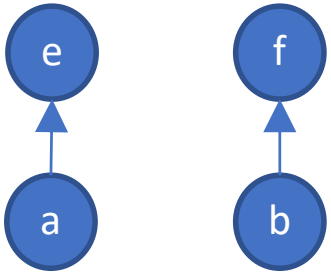




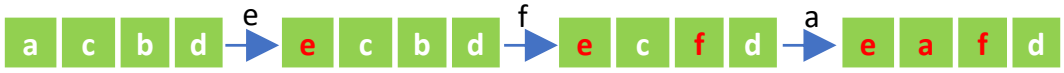
Chains, Explained

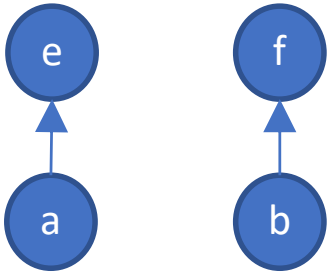


Chains, Explained

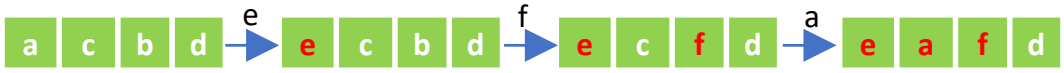


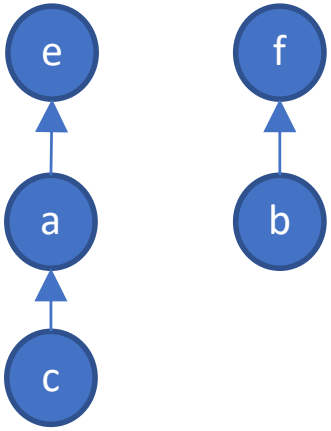
Chains, Explained



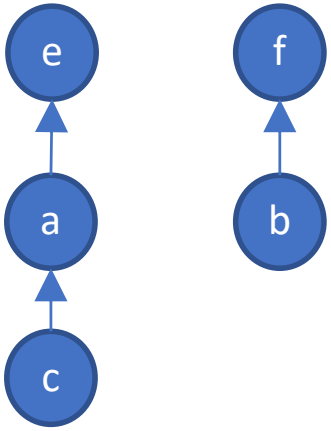


Chains, Explained

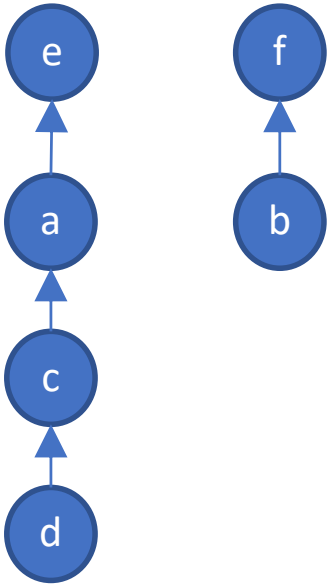




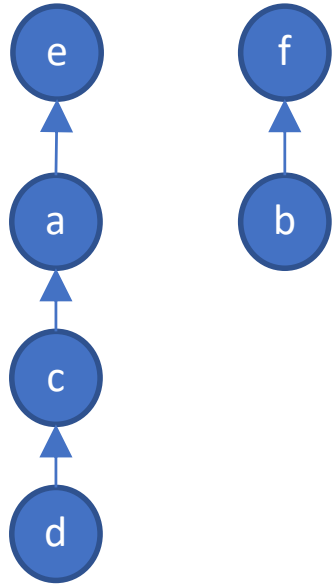
Chains, Explained



Chains, Explained

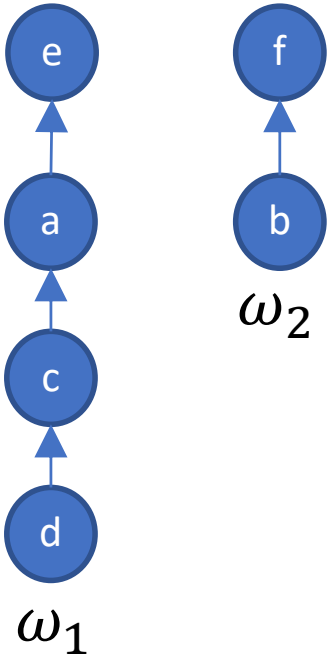


Chains, Explained

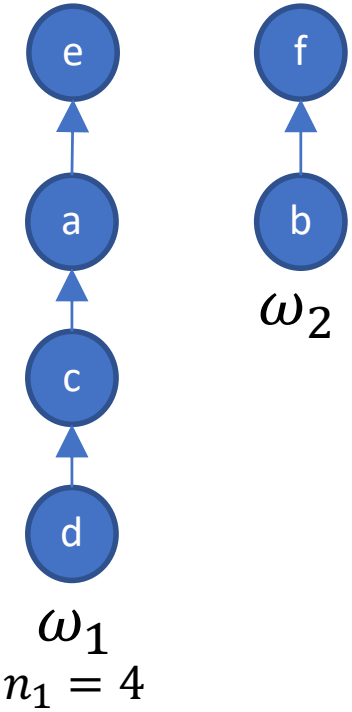


ω_1

Chains, Explained



Chains, Explained

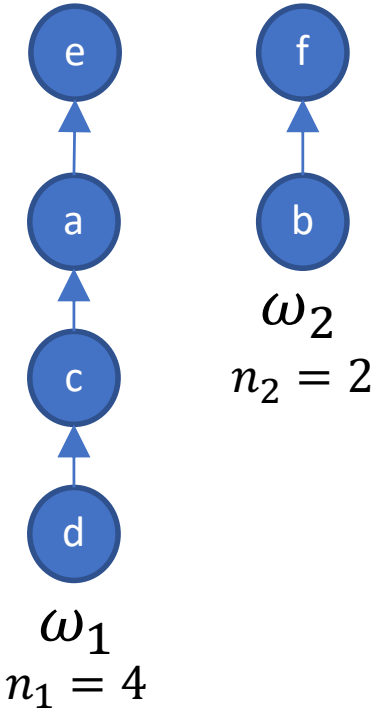


ω_1

$n_1 = 4$

ω_2

Chains, Explained



The Predictive Marker Algorithm

Now to the algorithm:

$\mathcal{PM}$ – *Predictive Marker*

The Predictive Marker Algorithm

Now to the algorithm:

$\mathcal{PM}$ – *Predictive Marker*

- At each phase, unmark all elements and save them as potentially **stale**.

The Predictive Marker Algorithm

Now to the algorithm:

$\mathcal{PM}$ – *Predictive Marker*

- At each phase, unmark all elements and save them as potentially **stale**.
- In a cache miss for z_i:

The Predictive Marker Algorithm

Now to the algorithm:

$\mathcal{PM}$ – *Predictive Marker*

- At each phase, unmark all elements and save them as potentially **stale**.
- In a cache miss for z_i:
  - If the element z_i is **clean**, create a new chain.

The Predictive Marker Algorithm

Now to the algorithm:

$\mathcal{PM}$ – *Predictive Marker*

- At each phase, unmark all elements and save them as potentially **stale**.
- In a cache miss for z_i:
  - If the element z_i is **clean**, create a new chain.
  - If z_i is **stale**, find its chain $z_i = \omega_c$.

The Predictive Marker Algorithm

Now to the algorithm:

$\mathcal{PM}$ – *Predictive Marker*

- At each phase, unmark all elements and save them as potentially **stale**.
- In a cache miss for z_i:
  - If the element z_i is **clean**, create a new chain.
  - If z_i is **stale**, find its chain $z_i = \omega_c$.
    - If the chain length is $n_c \leq H_k$:
      evict unmarked element with highest predicted time e.

The Predictive Marker Algorithm

Now to the algorithm:

$\mathcal{PM}$ – *Predictive Marker*

- At each phase, unmark all elements and save them as potentially **stale**.
- In a cache miss for z_i:
  - If the element z_i is **clean**, create a new chain.
  - If z_i is **stale**, find its chain $z_i = \omega_c$.
    - If the chain length is $n_c \leq H_k$:
      evict unmarked element with highest predicted time e.
    - Else:
      evict random unmarked element e.

The Predictive Marker Algorithm

Now to the algorithm:

$\mathcal{PM}$ – *Predictive Marker*

- At each phase, unmark all elements and save them as potentially **stale**.
- In a cache miss for z_i:
  - If the element z_i is **clean**, create a new chain.
  - If z_i is **stale**, find its chain $z_i = \omega_c$.
    - If the chain length is $n_c \leq H_k$:
      evict unmarked element with highest predicted time e.
    - Else:
      evict random unmarked element e.
    - Increase the chain: $n_c \leftarrow n_c + 1$, $\omega_c \leftarrow e$.

Main Theorem

Theorem

Consider the caching scenario, with the prediction model $\mathcal{H}$ and the loss function ℓ_1.
The competitive ratio is of the ϵ-assisted Predictive Marker Algorithm $\mathcal{PM}$ is bounded by:

$$\mathrm{CR}_{\mathcal{PM}}(\epsilon) \leq 2 \cdot \min\left(1 + \sqrt{5\epsilon}, 2H_k\right)$$

Proof - Chain Lengths

- Every cache miss is a link in a chain.

Proof - Chain Lengths

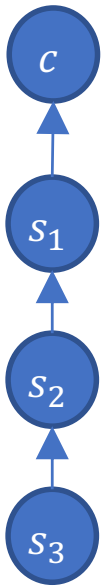
- Every cache miss is a link in a chain.
- Long chains means there are many misses. We would like to bound that length.

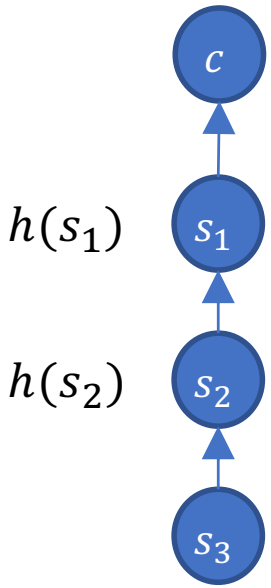
Proof - Chain Lengths

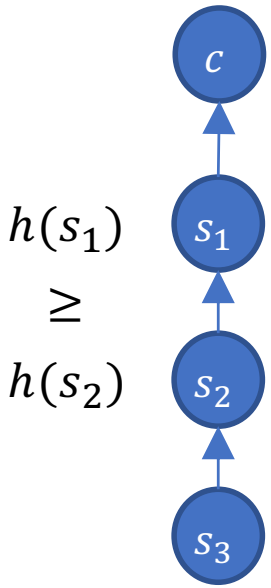
- Every cache miss is a link in a chain.
- Long chains means there are many misses. We would like to bound that length.

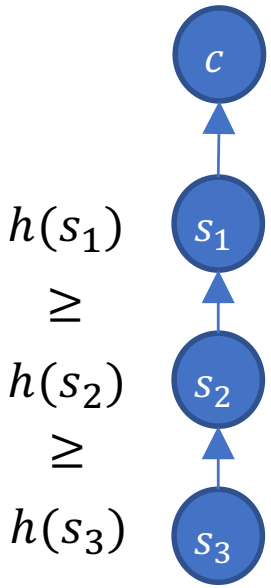
Lemma

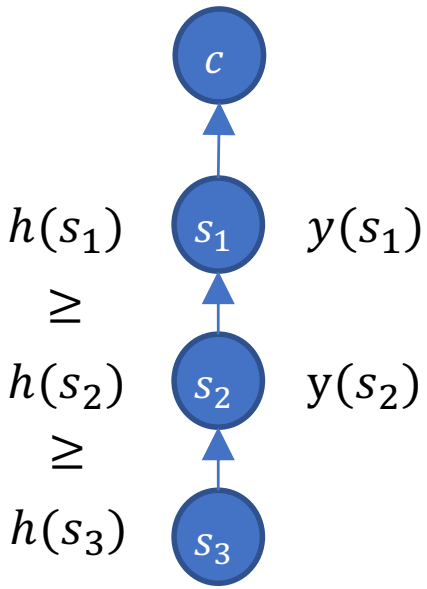
If h has error $\leq \eta$ on chain ω_c, then the chain's length is bounded by $n_c \leq 1 + \sqrt{5\eta}$.

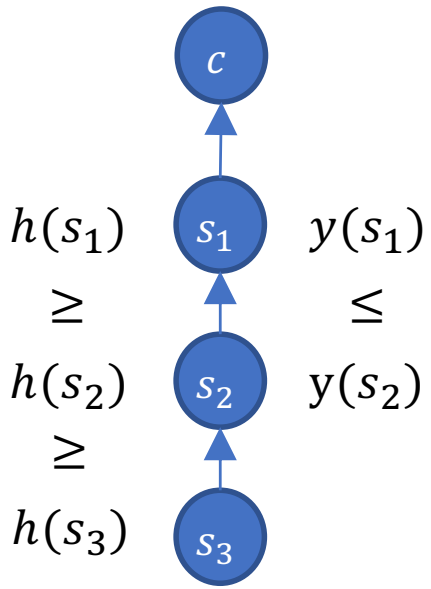

c
s_1
s_2
s_3

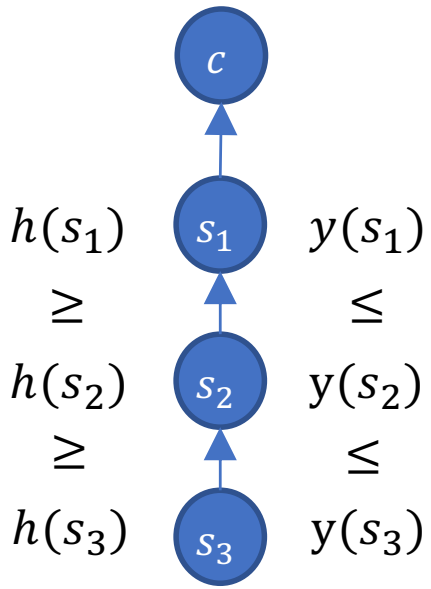

c
$h(s_1)$
s_1
$h(s_2)$
s_2
s_3

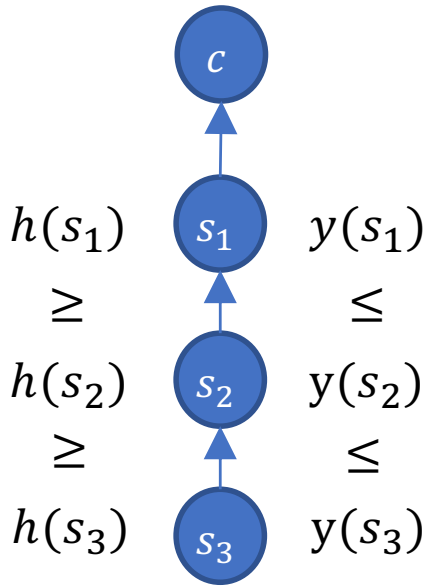

c
$h(s_1)$
s_1
$\geq$
$h(s_2)$
s_2
s_3


c
$h(s_1)$
s_1
$\geq$
$h(s_2)$
s_2
$\geq$
$h(s_3)$
s_3


c
$h(s_1)$
s_1
$y(s_1)$
$\geq$
$h(s_2)$
s_2
$y(s_2)$
$\geq$
$h(s_3)$
s_3


c
$h(s_1)$
s_1
$y(s_1)$
$\geq$
$\leq$
$h(s_2)$
s_2
$y(s_2)$
$\geq$
$h(s_3)$
s_3

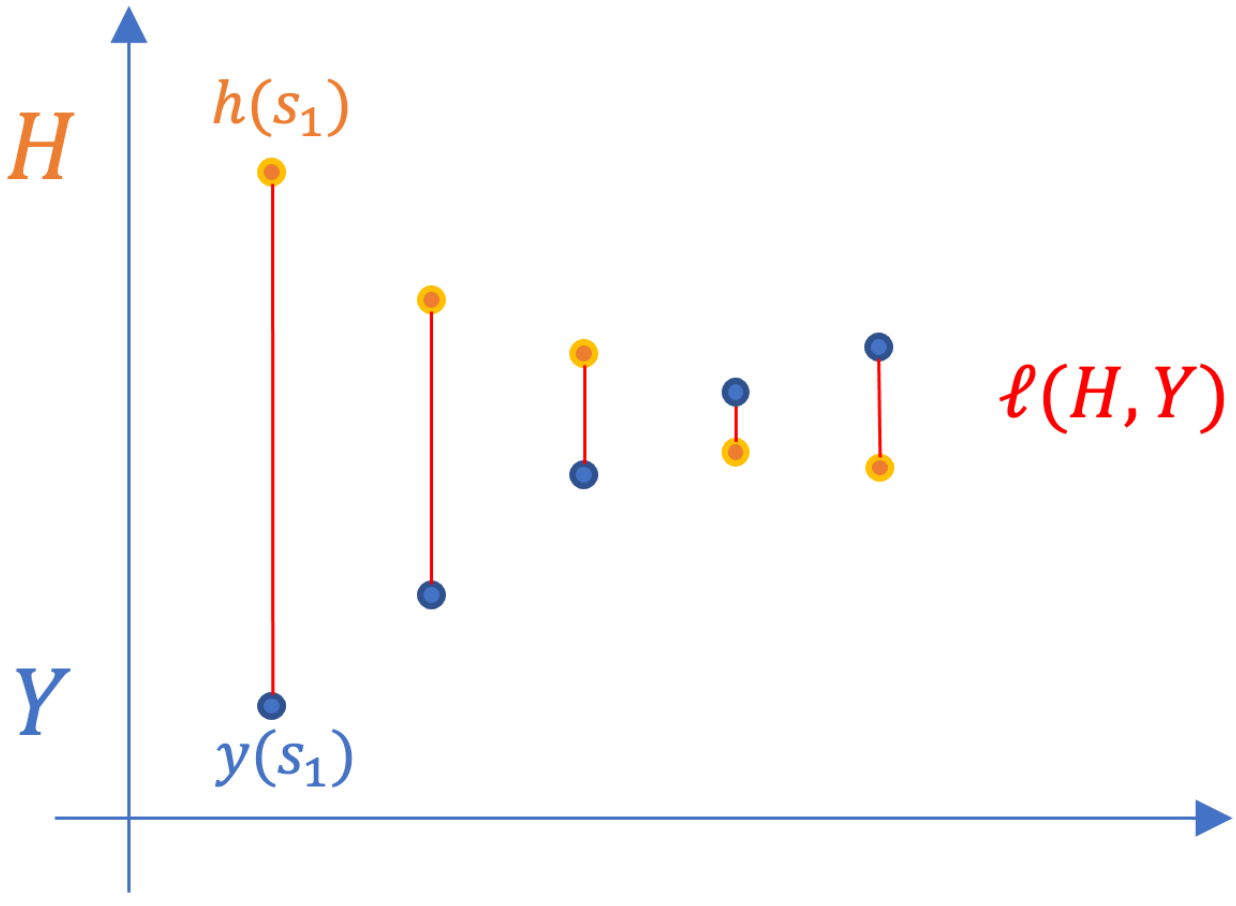

c
$h(s_1)$
s_1
$y(s_1)$
$\geq$
$\leq$
$h(s_2)$
s_2
$y(s_2)$
$\geq$
$\leq$
$h(s_3)$
s_3
$y(s_3)$



$$h(s_1) \geq h(s_2) \geq h(s_3) \geq \cdots$$

$$y(s_1) \leq y(s_2) \leq y(s_3) \geq \cdots$$

Proof - Chain Lengths

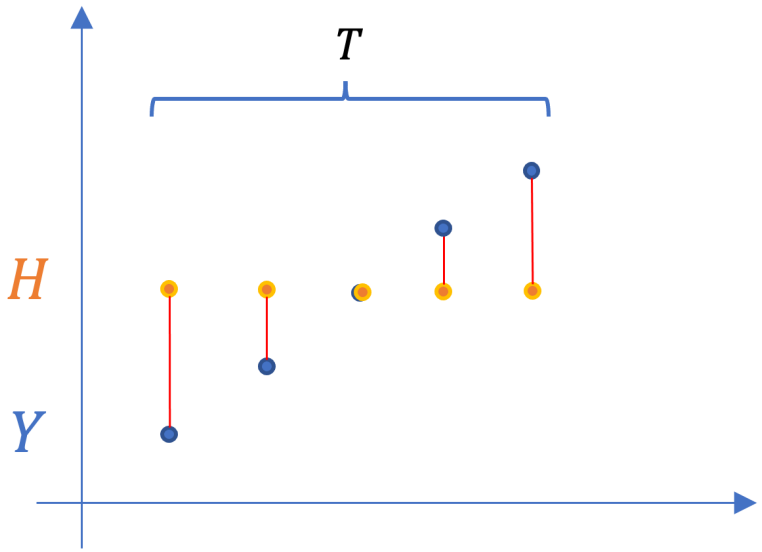


Proof - Chain Lengths

- If the error $\ell(H, Y)$ is bounded, how long can the chain get?

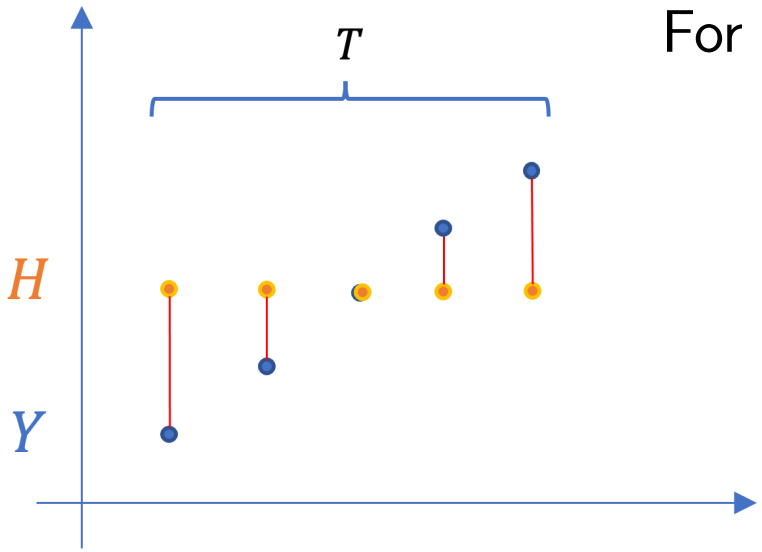
Proof - Chain Lengths

- If the error $\ell(H, Y)$ is bounded, how long can the chain get?



Proof - Chain Lengths

- If the error $\ell(H, Y)$ is bounded, how long can the chain get?



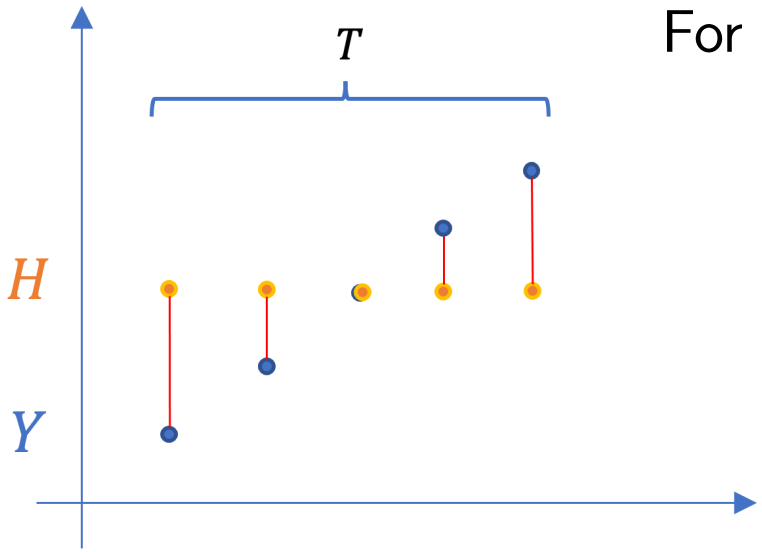
For such chain of length T we get the error:

$$\ell(H, Y) = 2\sum_{i=1}^{\frac{T-1}{2}} i = 2\frac{\frac{T-1}{2}\cdot\frac{T+1}{2}}{2}$$

$$= \frac{T^2-1}{4}$$

Proof - Chain Lengths

- If the error $\ell(H, Y)$ is bounded, how long can the chain get?



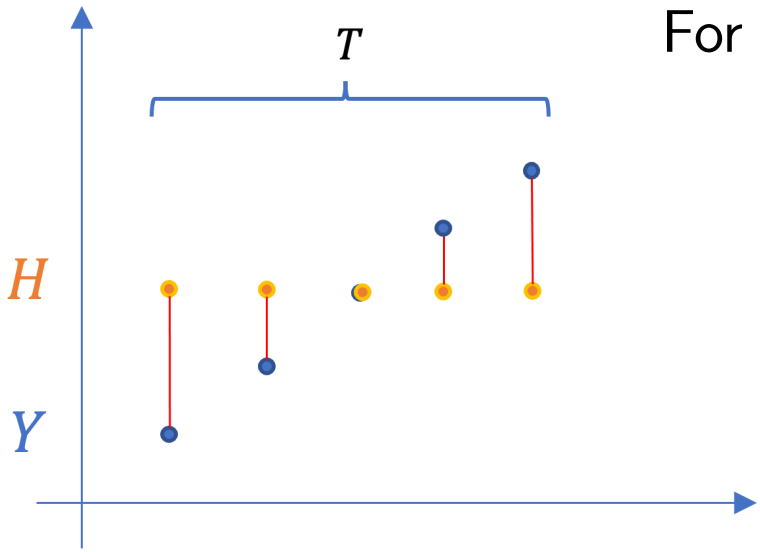
For such chain of length T we get the error:

$$\ell(H, Y) = 2\sum_{i=1}^{\frac{T-1}{2}} i = 2\frac{\frac{T-1}{2}\cdot\frac{T+1}{2}}{2}$$
$$= \frac{T^2-1}{4}$$

- So every chain of length T has at least $\frac{T^2-1}{4}$ error.

Proof - Chain Lengths

- If the error $\ell(H, Y)$ is bounded, how long can the chain get?



For such chain of length T we get the error:

$$\ell(H, Y) = 2\sum_{i=1}^{\frac{T-1}{2}} i = 2\frac{\frac{T-1}{2}\cdot\frac{T+1}{2}}{2}$$
$$= \frac{T^2-1}{4}$$

- So every chain of length T has at least $\frac{T^2-1}{4}$ error.

Corollary

If a chain has error $\leq \eta_c$, its length is at most $\sqrt{4\eta_c+1} \leq \sqrt{5\eta_c}$.

Proof - Continued

- We want to figure out the total cost.

Proof - Continued

- We want to figure out the total cost.

Proof - Continued

- We want to figure out the total cost.
- For a single chain c with error η_c:

Proof - Continued

- We want to figure out the total cost.
- For a single chain c with error η_c:
  - If we never switched to random evictions: there are at most $1 + \sqrt{5\eta_c}$ cache misses.

Proof - Continued

- We want to figure out the total cost.
- For a single chain c with error η_c:
  - If we never switched to random evictions: there are at most $1 + \sqrt{5\eta_c}$ cache misses.
  - If we did: there are at most $2H_k$ misses.

Proof - Continued

- We want to figure out the total cost.
- For a single chain c with error η_c:
  - If we never switched to random evictions: there are at most $1 + \sqrt{5\eta_c}$ cache misses.
  - If we did: there are at most $2H_k$ misses.
- We can bound the evictions from the chain c of the rth phase in expectation by $\min\left(1 + \sqrt{5\eta_{r,c}}, 2H_k\right)$.

Proof - Continued

- We want to figure out the total cost.
- For a single chain c with error η_c:
  - If we never switched to random evictions: there are at most $1 + \sqrt{5\eta_c}$ cache misses.
  - If we did: there are at most $2H_k$ misses.
- We can bound the evictions from the chain c of the rth phase in expectation by $\min\left(1 + \sqrt{5\eta_{r,c}}, 2H_k\right)$.
- So the total cost is bounded by $cost_{\mathcal{PM}(\epsilon)} \leq \sum_{r,c} \min\left(1 + \sqrt{5\eta_{r,c}}, 2H_k\right) \leq ?$

Proof - Continued

Proof.

Let L be the number of clean elements (= number of chains).

Proof - Continued

Proof.

Let L be the number of clean elements (= number of chains).

Proof - Continued

Proof.

Let L be the number of clean elements (= number of chains).
The total error is $\eta = \epsilon \cdot \textsc{Opt}$.

Proof - Continued

Proof.

Let L be the number of clean elements (= number of chains).
The total error is $\eta = \epsilon \cdot \text{OPT}$.
We want to know how big is $\sum_{r,c} \min\left(1 + \sqrt{5\eta_c}, 2H_k\right)$.

Proof - Continued

Proof.

Let L be the number of clean elements (= number of chains).
The total error is $\eta = \epsilon \cdot \textsc{Opt}$.
We want to know how big is $\sum_{r,c} \min\left(1 + \sqrt{5\eta_c}, 2H_k\right)$.

- Since $\sqrt{\cdot}$ and $\min(\cdot)$ are both concave:

Proof - Continued

Proof.

Let L be the number of clean elements (= number of chains).
The total error is $\eta = \epsilon \cdot \textsc{Opt}$.
We want to know how big is $\sum_{r,c} \min\left(1 + \sqrt{5\eta_c}, 2H_k\right)$.

- Since $\sqrt{\cdot}$ and $\min(\cdot)$ are both concave:
  - the way to maximize the expression is to divide the error η equally across all chains, with $\frac{\eta}{L}$ each.

Proof - Continued

Proof.

Let L be the number of clean elements (= number of chains).
The total error is $\eta = \epsilon \cdot \text{OPT}$.
We want to know how big is $\sum_{r,c} \min\left(1 + \sqrt{5\eta_c}, 2H_k\right)$.

- Since $\sqrt{\cdot}$ and $\min(\cdot)$ are both concave:
  - the way to maximize the expression is to divide the error η equally across all chains, with $\frac{\eta}{L}$ each.
- The total length of all chains is then $L \cdot \min\left(1 + \sqrt{5\frac{\eta}{L}}, 2H_k\right)$.

Proof - Continued

Proof.

Let L be the number of clean elements (= number of chains).
The total error is $\eta = \epsilon \cdot \textsc{Opt}$.
We want to know how big is $\sum_{r,c} \min\left(1 + \sqrt{5\eta_c}, 2H_k\right)$.

- Since $\sqrt{\cdot}$ and $\min(\cdot)$ are both concave:
  - the way to maximize the expression is to divide the error η equally across all chains, with $\frac{\eta}{L}$ each.
- The total length of all chains is then $L \cdot \min\left(1 + \sqrt{5\frac{\eta}{L}}, 2H_k\right)$.
- By Lemma 1, $\frac{L}{2} \leq \textsc{Opt}(\sigma)$. Trivially, $\textsc{Opt}(\sigma) \leq L$.

Proof - Continued

Proof.

Let L be the number of clean elements (= number of chains).
The total error is $\eta = \epsilon \cdot \text{OPT}$.
We want to know how big is $\sum_{r,c} \min\left(1 + \sqrt{5\eta_c}, 2H_k\right)$.

- Since $\sqrt{\cdot}$ and $\min(\cdot)$ are both concave:
  - the way to maximize the expression is to divide the error η equally across all chains, with $\frac{\eta}{L}$ each.
- The total length of all chains is then $L \cdot \min\left(1 + \sqrt{5\frac{\eta}{L}}, 2H_k\right)$.
- By Lemma 1, $\frac{L}{2} \leq \text{OPT}(\sigma)$. Trivially, $\text{OPT}(\sigma) \leq L$.
- So: $cost_{\mathcal{PM}(\epsilon)}(\sigma) \leq 2 \cdot \text{OPT}(\sigma) \cdot \min\left(1 + \sqrt{5\frac{\eta}{\text{OPT}(\sigma)}}, 2H_k\right)$.

Proof - Continued

Proof.

Let L be the number of clean elements (= number of chains).
The total error is $\eta = \epsilon \cdot \text{OPT}$.
We want to know how big is $\sum_{r,c} \min\left(1 + \sqrt{5\eta_c}, 2H_k\right)$.

- Since $\sqrt{\cdot}$ and $\min(\cdot)$ are both concave:
  - the way to maximize the expression is to divide the error η equally across all chains, with $\frac{\eta}{L}$ each.
- The total length of all chains is then $L \cdot \min\left(1 + \sqrt{5\frac{\eta}{L}}, 2H_k\right)$.
- By Lemma 1, $\frac{L}{2} \leq \text{OPT}(\sigma)$. Trivially, $\text{OPT}(\sigma) \leq L$.
- So: $cost_{\mathcal{PM}(\epsilon)}(\sigma) \leq 2 \cdot \text{OPT}(\sigma) \cdot \min\left(1 + \sqrt{5\frac{\eta}{\text{OPT}(\sigma)}}, 2H_k\right)$.

Proof - Continued

Proof.

Let L be the number of clean elements (= number of chains).
The total error is $\eta = \epsilon \cdot \mathrm{OPT}$.
We want to know how big is $\sum_{r,c} \min\left(1 + \sqrt{5\eta_c}, 2H_k\right)$.

- Since $\sqrt{\cdot}$ and $\min(\cdot)$ are both concave:
  - the way to maximize the expression is to divide the error η equally across all chains, with $\frac{\eta}{L}$ each.
- The total length of all chains is then $L \cdot \min\left(1 + \sqrt{5\frac{\eta}{L}}, 2H_k\right)$.
- By Lemma 1, $\frac{L}{2} \leq \mathrm{OPT}(\sigma)$. Trivially, $\mathrm{OPT}(\sigma) \leq L$.
- So: $cost_{\mathcal{PM}(\epsilon)}(\sigma) \leq 2 \cdot \mathrm{OPT}(\sigma) \cdot \min\left(1 + \sqrt{5\frac{\eta}{\mathrm{OPT}(\sigma)}}, 2H_k\right)$.

Which means $\mathrm{CR}_{\mathcal{PM}}(\epsilon) \leq 2 \cdot \min\left(1 + \sqrt{5\epsilon}, 2\log k\right)$. □

Tightness of analysis

Theorem (without proof)

Tightness of analysis

Theorem (without proof)

Any **deterministic** ϵ-assisted marking algorithm $\mathcal{A}$, that only uses the predictor in tie-breaking among unmarked elements in a deterministic fashion, has a competitive ratio of

$$\mathrm{CR}_{\mathcal{A}}(\epsilon) = \Omega\left(\min\left(\sqrt{\epsilon}, \mathbf{k}\right)\right)$$

Free parameter in the algorithm

- So far, we chose H_k as a switching point for the algorithm.

Free parameter in the algorithm

- So far, we chose H_k as a switching point for the algorithm.
- What about γH_k for some $\gamma > 0$?

Free parameter in the algorithm

- So far, we chose H_k as a switching point for the algorithm.
- What about γH_k for some $\gamma > 0$?

Theorem

Suppose that, for $\gamma > 0$, the algorithm uses γH_k as switching point. denote this algorithm by $\mathcal{PM}(\gamma)$.

Free parameter in the algorithm

- So far, we chose H_k as a switching point for the algorithm.
- What about γH_k for some $\gamma > 0$?

Theorem

Suppose that, for $\gamma > 0$, the algorithm uses γH_k as switching point. denote this algorithm by $\mathcal{PM}(\gamma)$.
Then, the competitive ratio of ϵ-assisted ϵ-assisted $\mathcal{PM}(\gamma)$ is bounded by:

$$CR_{\mathcal{PM}(\gamma),\ell}(\epsilon) \leq 2 \min\left(1 + \frac{1+\gamma}{\gamma}\sqrt{5\epsilon}, (1+\gamma) H_k, k\right)$$

Free parameter in the algorithm

- So far, we chose H_k as a switching point for the algorithm.
- What about γH_k for some $\gamma > 0$?

Theorem

Suppose that, for $\gamma > 0$, the algorithm uses γH_k as switching point. denote this algorithm by $\mathcal{PM}(\gamma)$.
Then, the competitive ratio of ϵ-assisted ϵ-assisted $\mathcal{PM}(\gamma)$ is bounded by:

$$CR_{\mathcal{PM}(\gamma),\ell}(\epsilon) \leq 2 \min\left(1 + \frac{1+\gamma}{\gamma}\sqrt{5\epsilon}, (1+\gamma) H_k, k\right)$$

- What does a low γ mean?

Free parameter in the algorithm

- So far, we chose H_k as a switching point for the algorithm.
- What about γH_k for some $\gamma > 0$?

Theorem

Suppose that, for $\gamma > 0$, the algorithm uses γH_k as switching point. denote this algorithm by $\mathcal{PM}(\gamma)$.
Then, the competitive ratio of ϵ-assisted ϵ-assisted $\mathcal{PM}(\gamma)$ is bounded by:

$$CR_{\mathcal{PM}(\gamma),\ell}(\epsilon) \leq 2 \min\left(1 + \frac{1+\gamma}{\gamma}\sqrt{5\epsilon}, (1+\gamma) H_k, k\right)$$

- What does a low γ mean?
- What does a high γ mean?

Robustifiying LRU

- How can we use LRU in the predictive marker setting?

Robustifiying LRU

- How can we use LRU in the predictive marker setting?
- Define $h(\sigma_i) = -i$.

Robustifiying LRU

- How can we use LRU in the predictive marker setting?
- Define $h(\sigma_i) = -i$.

Fact

If we never switched to random evictions, this is exactly LRU.

Robustifiying LRU

- How can we use LRU in the predictive marker setting?
- Define $h(\sigma_i) = -i$.

Fact

If we never switched to random evictions, this is exactly LRU.

- LRU is deterministic and therfore has only a bound of $\Theta(k)$, but is very good in practice.

Robustifiying LRU

- How can we use LRU in the predictive marker setting?
- Define $h(\sigma_i) = -i$.

Fact

If we never switched to random evictions, this is exactly LRU.

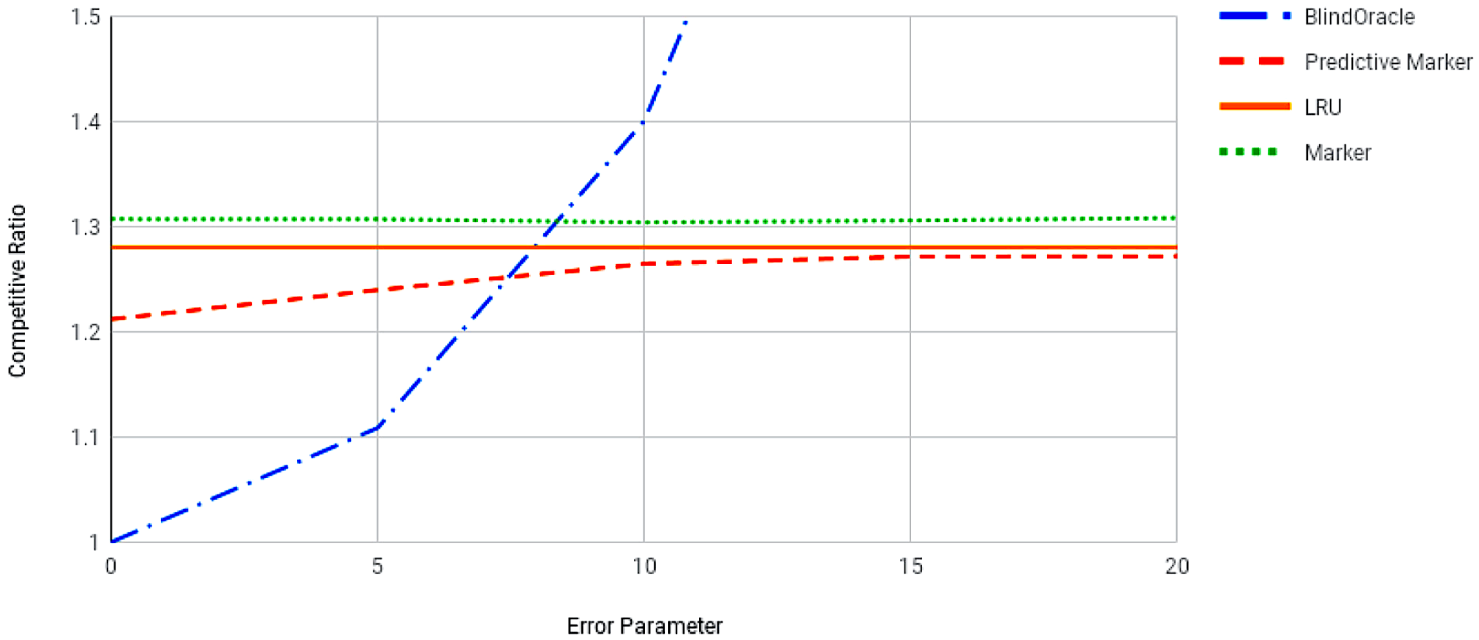
- LRU is deterministic and therfore has only a bound of $\Theta(k)$, but is very good in practice.
- This new setting reduces the analysis of LRU from $\Theta(k)$ to $O(\log k)$, while still exploiting its predictive power.

Comparing results

- Lets take a look at some real world datasets:

Dataset	Num Sequences	Sequence Length	Unique Elements
BK	100	2,101	67– 800
Citi	24	25,000	593 – 719

Comparing results



Algorithm	Competitive Ratio on BK	Competitive Ratio on Citi
Blind Oracle	2.049	2.023
LRU	1.280	1.859
Marker	1.310	1.869
Predictive Marker	1.266	1.810

- We saw a general framework for combining **online algorithms** with guidance of **machine learning** - OMLA.

- We saw a general framework for combining **online algorithms** with guidance of **machine learning** - OMLA.
- We saw how we can maintain **robustness** while exploiting predictions to improve **competitiveness**.

- We saw a general framework for combining **online algorithms** with guidance of **machine learning** - OMLA.
- We saw how we can maintain **robustness** while exploiting predictions to improve **competitiveness**.
- We discussed the analysis of **robustness vs competitiveness** and looked at real-world examples.

- We saw a general framework for combining **online algorithms** with guidance of **machine learning** - OMLA.
- We saw how we can maintain **robustness** while exploiting predictions to improve **competitiveness**.
- We discussed the analysis of **robustness vs competitiveness** and looked at real-world examples.

- Thank you!

 Thodoris Lykouris and Sergei Vassilvitskii.
2021. Competitive Caching with Machine Learned Advice.
J. ACM 68, 4, Article 24 (July 2021), 25 pages.